

**BEIS**  > Subsidy Control Project > Test Strategy

|  |  |
| --- | --- |
|  |  |
| Subsidy Control – Transparency Database Test Strategy | | |
| Subsidy Control Project | |
| August 2020 | |

Contents

[Document Control 5](#_Toc48357516)

[Document Revision History 5](#_Toc48357517)

[Distribution 5](#_Toc48357518)

[1. Introduction 6](#_Toc48357519)

[2. Executive Summary 7](#_Toc48357520)

[3. Subsidy Control UK Transparency Database Overview 8](#_Toc48357521)

[4. Document Overview 9](#_Toc48357522)

[4.1. Purpose 9](#_Toc48357523)

[4.2. Scope of this document 9](#_Toc48357524)

[4.3. Domain Architecture Overview 10](#_Toc48357525)

[4.3.1. Technical Architecture Identification 10](#_Toc48357526)

[4.4. Related Documentation 11](#_Toc48357527)

[5. Test Process 12](#_Toc48357528)

[5.1. Feature grooming 12](#_Toc48357529)

[5.2. Sprint Planning 12](#_Toc48357530)

[5.3. Sprint Testing Activities 13](#_Toc48357531)

[5.3.1. Test Planning 13](#_Toc48357532)

[5.3.2. Test data preparation 13](#_Toc48357533)

[5.3.3. Test design 13](#_Toc48357534)

[5.3.4. Test execution 13](#_Toc48357535)

[5.3.5. Test reporting 14](#_Toc48357536)

[5.4. Team demo 14](#_Toc48357537)

[5.5. Sprint review/retrospective 14](#_Toc48357538)

[5.6. Testing Pre-requisites 14](#_Toc48357539)

[5.7. Assumptions 14](#_Toc48357540)

[5.8. Dependencies 15](#_Toc48357541)

[6. Test Methodology 16](#_Toc48357542)

[7. Test Approach 16](#_Toc48357543)

[8. Testing Stages 18](#_Toc48357544)

[8.1. Unit Testing 18](#_Toc48357545)

[8.2. Acceptance testing (AT) 19](#_Toc48357546)

[8.2.1. Portal frontend – GUI 19](#_Toc48357547)

[8.2.2. Experience API, Process and System APIs 20](#_Toc48357548)

[8.3. Database testing: 21](#_Toc48357549)

[8.4. Exploratory testing 21](#_Toc48357550)

[8.5. End to End Testing (E2E) 22](#_Toc48357551)

[8.6. User Acceptance Testing (UAT / Private Beta) 23](#_Toc48357552)

[8.7. Performance Testing 24](#_Toc48357553)

[8.8. Security Testing 26](#_Toc48357554)

[8.9. Accessibility testing 27](#_Toc48357555)

[8.10. Operational Acceptance Testing (OAT) 28](#_Toc48357556)

[9. Public Beta 30](#_Toc48357557)

[10. Definition of Ready and Definition of Done 30](#_Toc48357558)

[10.1. Definition of Ready 30](#_Toc48357559)

[10.2. Definition of Done 30](#_Toc48357560)

[11. QA and automation tools 31](#_Toc48357561)

[12. Environments 32](#_Toc48357562)

[13. Defect Management 32](#_Toc48357563)

[13.1.1. Defect Priority: 36](#_Toc48357564)

[14. Status Reporting & Meetings 36](#_Toc48357565)

[15. Testing Governance 37](#_Toc48357566)

[16. Suspension Criteria 37](#_Toc48357567)

[17. Resumption Criteria 38](#_Toc48357568)

[18. Risks Management 38](#_Toc48357569)

[19. Escalation Procedures 39](#_Toc48357570)

[20. Quality Gates 39](#_Toc48357571)

[18. Skills Required 41](#_Toc48357572)

[19. Appendix (To be updated) 42](#_Toc48357573)

[19.1. Acronyms and Abbreviations 42](#_Toc48357574)

[19.2. Terms 42](#_Toc48357575)

# 

# Document Control

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Title** | Subsidy Control – Transparency Database Test Strategy | | |
| **Version** | Draft | | |
| **Author(s)** | Vimalnath N | | |
| **Authorized by** |  | Date |  |
| **Approvers (optional)** |  | Date |  |

# Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Revision Date** | **Changed by** | **Summary of Changes** |
| First draft submission | 03/08/2020 | Vimalnath N | Initial draft of the Test strategy |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Distribution

This document has been distributed to:

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Individual/Team/Distribution List** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Introduction

For overall context, the goal of the Testing work stream on the Subsidy Control Transparency Database (hereinafter referred as Transparency DB) project is to create and document the testing approach and plan for the “To Be” solution. This document proposes the strategy, approach and procedures for the support, direction and execution of all testing phases for the “To Be” Solution in the Beta phase of the Transparency DB project. The success of the project is dependent upon delivering a system that has been designed and implemented based on BEIS user requirements, priorities and expectations. The development and use of a comprehensive testing strategy will support this goal. This testing strategy is intended for both system users and management to provide an understanding of and overall view into the different testing phases and processes.

# Executive Summary

The strategy and approach to testing is a critical contribution to the successful implementation of the “To Be” solution.  The approach taken must ensure a comprehensive understanding of the risk profile and how each stage of testing will be performed. The proposed approach should also document roles and responsibilities, opportunities to automate testing, alignment to the Government digital standards that will be used throughout the testing process.

The overall objective of the Test Strategy is to ensure and formally document that the “To Be” Solution meets the business requirements of the Transparency DB project. This includes testing of the following areas:

1. The developed system functionality of the overall solution including user journeys, business processes and process steps
2. Integration with other systems and tools (as applicable)
3. End user security and authorisations for GUI[[1]](#footnote-1) & APIs
4. Testing of non-functional requirements for the “To Be” Solution

# Subsidy Control UK Transparency Database Overview

BEIS (Business, Enterprise and Industrial Strategy) department is making the UK a great place to work and do business. Their priorities include maximising investment opportunities and bolstering UK interests as we leave the EU and promote competitive markets and responsible business practices. BEIS Subsidy Control team has been developing proposals for a bespoke domestic subsidy control regime not based on EU rules. This will amongst other things aim to bear down on anti-competitive subsidies through transparency about subsidies awarded.

The primary objective of the Transparency DB is to come up with a UK-specific transparency DB that will succeed the EU’s Transparency Aid Module (TAM) – the current system on which aid grantors must record all the individual aid awards of over EUR 500,000, within six months of the award being made.

Transparency in the grant's management process is viewed as the best way to ensure that competitiveness is maintained. Ensuring that the public have access to the grants that are awarded allows the public to scrutinise through Public User search of the Transparency Database.

# Document Overview

## Purpose

The purpose of this document is to propose a **high level** Test Strategy for the Transparency DB project. This document describes key components such as the types of testing, levels of testing, overarching test stages, responsibilities for each testing stage and the overall [quality assurance](https://www.gov.uk/service-manual/technology/quality-assurance-testing-your-service-regularly) approach that are aligned to the GDS principles. This document is intended to be used as a reference for all project team members. This includes the testing team, development team, business analysts, programme management team, business team and operations team.

A detailed Test Plan will be incorporated with each project release. This will contain granular information on the testing approach for the prescribed testing methods mentioned in this document.

## Scope of this document

The scope of this document is limited to the testing activities required for the “To Be” solution that is being implemented for the Transparency DB. At the time of drafting the test strategy, the Transparency DB is identified to be a standalone system with no integrations with TAM or any other applications available within the current BEIS IT landscape. Hence, the scope of the supplier’s testing activities shall focus on the following:

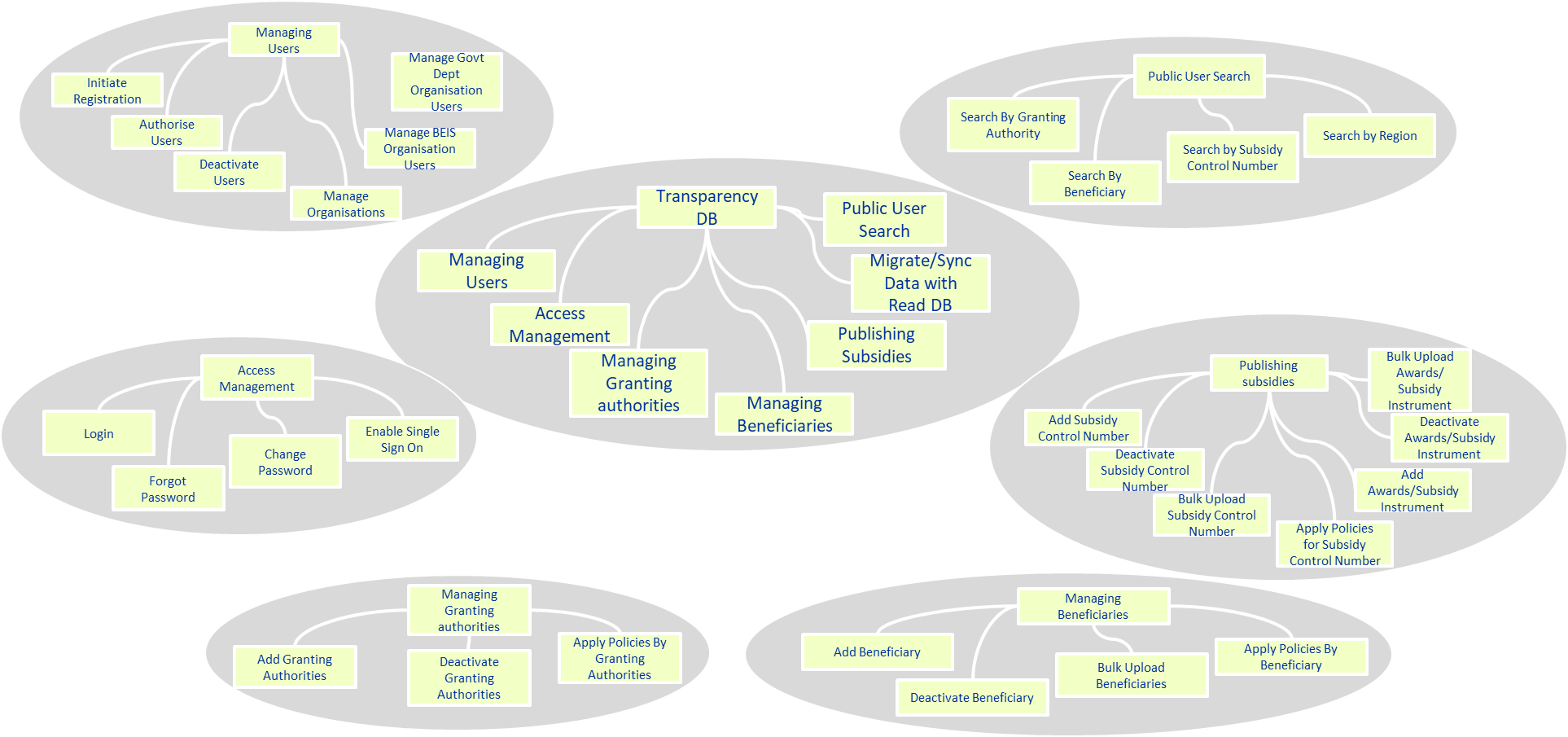
* Transparency Portal:
  + Internal:
    - Access management
    - Managing users
    - Single upload
    - Bulk upload
      * Publishing subsidies
      * Managing Granting authorities
      * Managing beneficiaries
      * Subsidy control IDs
    - Experience, Process and system APIs
    - Authentication and authorisations
    - Role Based Access Controls (RBAC)
    - Synchronisation of data with Read database
  + Public facing portal:
    - Public user search

Specific strategies and approaches to testing each system component will not be defined in this document and will be defined at a later stage. The strategy herein adheres to GDS standards and is designed to be flexible when required while still maintaining alignment. Activities can be performed in parallel where this has been agreed and is necessary to achieve deadlines or accelerated milestones.

Note, that some stages of testing may require separate teams and may not be completed by the supplier and such exclusions are noted as part of the ‘Out of scope’ section of this document.

## Domain Architecture Overview

The functional architecture of the “To Be” solution has been proposed to be built using the ‘Domain driven architecture’ that is aligned to the GDS standards. This model consists of system domains viz. Managing users, Public User search, Transparency DB, Managing beneficiaries, Publishing subsidies and Access management. The diagram below presents an illustrative high level domain view from a UK Transparency DB perspective.



***Figure 1-*** *illustrative domain architecture diagram*

### Technical Architecture Identification

At the time of writing the Test Strategy, two technical architecture solution options were proposed towards the end of the discovery phase. These options are being evaluated in the Alpha phase. The below table provides a high level view of the technical architecture that are being evaluated:

| **Component** | **Technology stack** | |
| --- | --- | --- |
| **Solution option 1** | **Solution option 2** |
| Digital UI Portal | React JS | React JS |
| API Gateway | Azure API Gateway | Zuul API Gateway |
| Experience APIs | Spring Boot | Spring Boot |
| Process APIs | Spring Boot + GraphQL | Camunda Workflow engine |
| Rules Engines | Drools | Camunda Decision Engine |
| Pub/Sub(Integration) | Spring Integration | Spring Integration |
| Database | PostGreSQL | PostGreSQL |
| Layer 7 firewall | ? | ? |
| User Management and RBAC | IdAM (Okta) | IdAM (Okta) |
| Notification services |  |  |
| Messaging Backbone |  |  |
| Identity Management (Open LDAP) |  | IdAM |

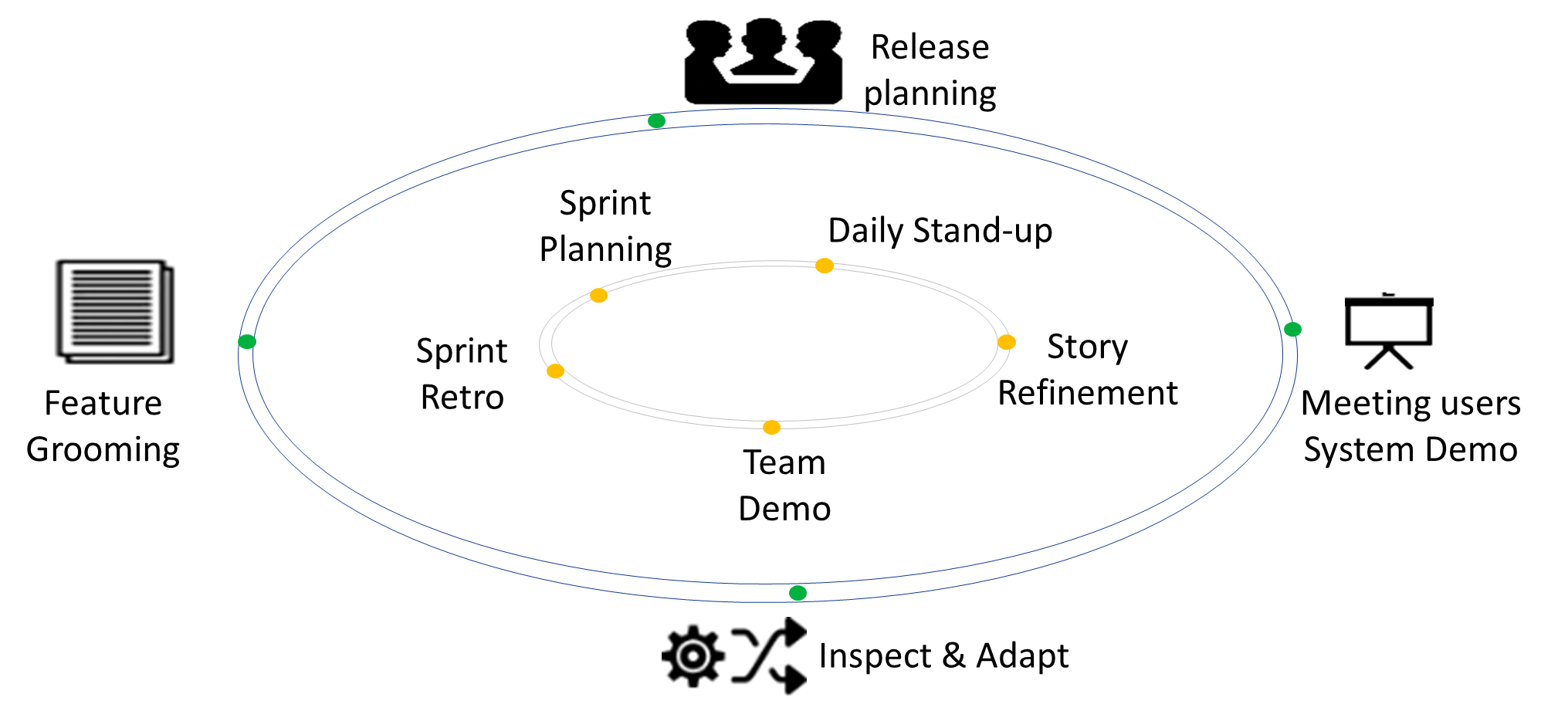
## Related Documentation

Throughout the Transparency database definition, design and accelerated build phase in the Beta phase, below documents would be produced to support the overall project which has a direct impact to this testing strategy. The table below contains a list of important documents that are proposed to be produced by the UI/UX, Technical design and development teams during the BETA phase that relate to the testing activities.

|  |  |
| --- | --- |
| **Document** | **Description** |
| User stories | Provides a description of the user needs, features needed and acceptance criterions written in a BDD (Given, When, Then) format |
| UI Wireframes | This document acts as a baseline for the user interface that will be created for the Transparency DB. These cover features, functions and content that must be presented in the UI |
| High Level Solution Design | This design document gives detail of the system architecture and database design for the “To Be” solution. It also provides a relation between various modules, system functions, data flow, flow charts and data structures. |
| Low level solution design | This document captures the detailed login for each and every component, class diagrams, program specs and relations between classes. |

# Test Process

The Agile QA process follows the agile sprint approach (refer picture below) enabling testing is done more often while incrementally building the Transparency DB in the BETA phase. The below sections outline the agile test process that shall be followed to ensure testing is done efficiently without impacting or compromising the test quality / outcomes.



## Feature grooming

This is the phase where the features written in the form of user stories by the Supplier’s business analyst along with acceptance criterions per story are reviewed. The BEIS product owner (PO) and the rest of the supplier’s team review if the backlog items contain sufficient information that can then be taken into the sprint planning and estimation following prioritisation by the BEIS PO. During this stage, any clarifications on how the story should function will be raised and discussed, any technical challenges noted and acceptance criterions validated for testability. This refinement helps provide more accurate estimates during the sprint planning sessions.

## Sprint Planning

Sprint planning phase is typically split into ‘Sprint Scoping’ and ‘Sprint Plan’. As part of the scoping exercise, the prioritised user stories are discussed within the sprint team (BEIS PO, supplier’s business analyst, UI/UX members, architects, developers and testers) to fully understand the user story and provide estimates. Based on the supplier team’s capacity and confidence to accomplish the work items within a sprint, the sprint backlog is created and sprint goal is established.

In the Beta phase, it is assumed a 2 week sprint cycle shall be followed as in the Alpha phase. The supplier team (Business Analyst, UI/UX members, architects, developers and testers) identify tasks for the backlog items needed to deliver the sprint user stories.

## Sprint Testing Activities

### Test Planning

At the time of sprint planning, the Supplier QA lead shall gain an understanding of the requirements, ensure test capacity / estimates are accounted for and appropriate risks, issues and dependencies are highlighted. Following this the supplier BA conducts story kick-off meetings to develop a common understanding between the developers and the QA team. Following this, the supplier tester team starts to identify the test scenarios (e.g. the user will be able to login with a valid username and password) and work closely with the supplier BA & developers to review the test scenarios or make changes based on code changes involved.

### Test data preparation

Following the review of the test scenarios, the supplier tester team starts preparing the test data needed to effectively test the user stories both manually and via automation. The testing team ensures different sets of data are used depending on the nature of the test being carried out. The supplier test team may create automated test data scripts that helps in bulking up test data to carry out various types of tests needed to ensure all planned tests can be completed.

### Test design

The supplier test team shall develop the sprint test cases using Behaviour Driven Development methodology (further details provided in Section 6 of this document). The primary reason for this being the acceptance criterions for the user stories are written using ‘given-when-then’ syntax (i.e. Gherkin language). This syntax allows automation tools like Cucumber, Serenity that targets a BDD approach to readily use these to create feature files for automation scripts.

### Test execution

Prior to the start of sprint test execution, the supplier’s developer(s) schedule a quick demo of the of the user story that has been developed with the supplier’s BA and a member of the test team to showcase the story along with the acceptance criterions has been developed to the business requirements. Following the demo, the user stories are deployed using a CI pipeline into the ‘Staging’ environment where all test execution activities shall be performed.

Following a new deployment trigger via the CI pipeline, regression test suites would be automatically triggered to ensure there are no discrepancy between environments. Following a successful run, all planned tests for the sprint would be executed by the supplier’s test team. Once all planned tests are complete, the supplier’s test team should conduct at-least a round of ‘Exploratory testing’ of the features around the changes developed to provide additional confidence there are no issues with the delivered solution. All defects identified during execution shall be raised as bugs and linked back to the tests to establish traceability.

Based on the time available within the sprint, the supplier’s test team may conduct ‘Early exploratory tests’ on a developer sand-box that enables ‘Fail fast and early strategy’ – a key principle in Agile.

### Test reporting

Following the completion of the planned tests, the test results shall be made captured in the test management tool e.g. Azure test plan. All tests shall be evaluated for exit criterions and any learnings from the sprint shall be captured during the sprint retrospective meetings.

## Team demo

Once all planned sprint and exploratory tests are complete, the supplier’s development team or business analyst shall demo the new sprint functionality to the product owner. Once BEIS PO or relevant business stakeholder confirms the solution complies with the acceptance criterions, the user stories are moved to an ‘Approved’ or ‘Done’ stage.

## Sprint review/retrospective

Following completion of a sprint, the sprint team (BEIS PO and Supplier’s team) assesses against Sprint Goal setup at the sprint planning session. Following this a sprint retrospective is conducted to inspect itself and any lessons learnt are captured to create a plan for improvements to be enacted during the next sprint.

## Testing Pre-requisites

Although each stage of testing will have an entry criteria, the following testing pre-requisites should be in place before any form of testing begins.

* There will be specific test environments (viz. Staging environment) made available to facilitate testing activities taking place across each testing stage
* The Staging test environment will be available with the latest code deployed via a CI pipeline along with initial base data setup
* Database access and appropriate permissions will be provided to the supplier’s testing team and any other individuals who require access
* All entry and exit criteria for each test stage must be reviewed and approved before the next testing stage can begin

## Assumptions

The assumptions and dependencies below have been identified to keep the testing process within the given project timeline. The assumptions and dependencies within this section relate solely to the test strategy.

* All test scripts and scenarios will be developed by the supplier’s QA team as per the test standards and will be completed prior to the test stage for which they are required.
* For test data setup, the supplier’s development team will provide the testing team with the sample files or a format template used to load the data into the system. The supplier’s test team will be responsible for the creation of test data in the staging environments for their test consumption
* All supplier’s developers, testers shall have access to relevant task management, test and defect management tools
* Data migration from existing TAM system to the Transparency database is considered out of scope and hence data migration testing is not applicable
* Security testing / Penetration testing will be performed by BEIS Security team or an approved third party service provider nominated by BEIS
* Any data needed for User Acceptance tests would be created by BEIS UAT testers prior to the start of testing

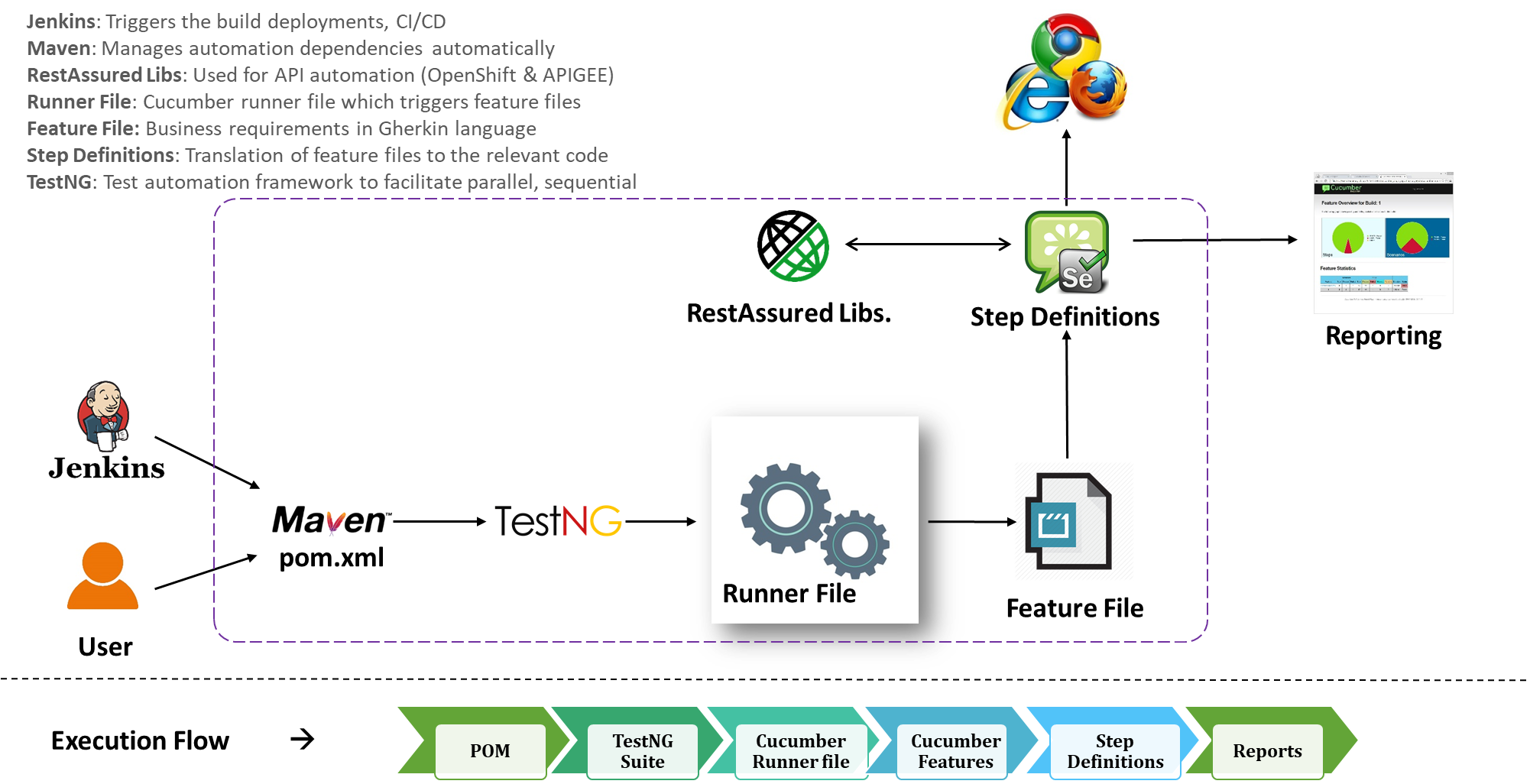
## Dependencies

* **Resources:** A sufficient number of dedicated, trained resources (from the Supplier) will be available to support all testing requirements for the BETA phase
* **Access:** All supplier test members will have access to the appropriate applications and other software necessary to complete the testing processes
* **System environments:** The appropriate system environments must be established before the testing process begins.
* **Data:** There is sufficient amount of new data defined and made available by the supplier’s development/test teams. Test data should be defined in terms of the Physical and Logical Data Models in the solution designs and a data dictionary produced.

# Test Methodology

Behaviour Driven Development (BDD) encourages communication between BEIS stakeholders (e.g. Product Owner, business teams) and Supplier’s teams so all members understand each feature, prior to the development process. In BDD, the acceptance criterions to the user scenarios are written in a specific format, the Gherkin ‘Given-When-Then’ syntax. This syntax can directly be used into feature files to explain how a feature behaves in different situations with varying input parameters. During the BETA phase, the supplier’s QA team shall establish the automation framework using BDD methodology along with relevant tools required to deliver the various tests viz. UI automation, RestAssured API tests etc.

BDD at a high level typically creates test to fail followed by passing the test scenario. A sample BDD Cucumber automation test framework architecture for the BETA phase is represented below. The supplier’s QA team shall review this at the start of the BETA phase.



*Illustrative Test Automation framework*

# Test Approach

The supplier’s testing tasks will be conducted in line with the Behaviour Driven Development (BDD) methodology. The supplier’s testers should be able to write tests based on the business behavior and the acceptance criterions found within the user stories even before the actual services are built. Following this approach allows the test scenarios to be identified and written before actual work starts and hence gives the BEIS PO & the Supplier’s BA enough time to review and if needed make or suggest any changes to the user stories. In such cases, the acceptance criterions would be refined in a collaborative fashion within the sprint team (BEIS PO and Supplier’s teams).

To achieve the above, acceptance criteria/scenarios will be written using the Gherkin syntax of Given, When and Then statements.

*E.g. Given I have a list of users with their respective email addresses*

*When I Bcc to bulk email them with a link to registration pages*

*Then nominated users receive their registration emails*

The acceptance criterions written in plain English live completely separately from the automation test code in the form of ‘Feature files’. These feature files are thereby easy to understand by both technical and non-technical members. This together with the test plan will form all the documentation for testing that can be provided to the business to verify the scenarios that allows easy traceability back to the user story or acceptance criteria. For each sprint, the Supplier’s test team shall ensure the test scenarios (automated and manual) are documented. These will be identified from the scope items in each sprint and include interrelated modules and components of the service that will be affected by the scope items. All test cases (manual and automated) shall be stored in the same location, making it easier to view what tests are captured.

Test automation is key to deliver Quality@speed and hence all feasible tests should be targeted for automation by the supplier’s test teams. These tests once stable should be integrated into the CI pipeline once a new feature has been tested and approved. These automated tests will be run on the Azure DevOps pipeline. All test scenarios shall be documented and tagged with test specific tags and version controlled as well. This will ensure all tests are documented and captured even if the tests cannot be automated. The supplier’s test team may wish to run automated tests as and when they are needed and it will be up to the team to decide what tests they may want to run with every code check-in, or what tests should be scheduled to run overnight. The tests will be executed against a given environment i.e. staging as this will ensure the test teams to know the current state of the environment and the systems, catching any issues at the earliest stage possible.

It is the responsibility of the supplier’s test teams to ensure the tests are built with the needed dependencies using appropriate tools e.g. Maven to make running and packaging of the tests as easy as possible. The aim is to continuously test that helps determine the quality of the application under development at any point in time. Continuous testing encourages early feedback and ensures all test failures are spotted and rectified at the earliest possible stage. The focus of this approach is to shorten the feedback loop and incrementally enhance the features and quality of the product.

To summarize, the objective of this test approach is to ‘Fail fast and early’ defect detection that helps reduce cost of quality, improvise the service and fix defects early in the test lifecycle.

# Testing Stages

## Unit Testing

Unit testing ensures that each systems/tools code is error free and that each program operates as specified. At this level of testing, supplier’s developers concentrate on the functional and technical quality of the unit.  More specifically, the objective of unit testing is to verify that the unit:

* Meets the functional requirements and that the actual execution of the unit matches the technical design
* Conforms to the Government standards in place;  
  Is consistent with the database and any other interfaces

The detailed test activities performed during this phase are illustrated in the table below:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Owner | * Supplier’s development team |
| Objective | * Validate that the individual functions are configured and/or developed to appropriately translate technical and functional business requirements of BEIS |
| Entry Criteria | * Initial solution design completed * Solution release note including feature set, known issues and limitations * Development environment available with user ID’s and required authorisations * Defect process has been identified * Solution configuration & build completed to enable Unit Testing start * Test data exists * Development tools set up |
| Activities | * Each individual configuration element will be tested * Each completed custom development object will be tested |
| Environment | * Development environment |
| Test data | * Mock data created by supplier’s developers |
| Tool | * IDE – Eclipse * Source control – GitHub * CI – Jenkins * Build tool – Maven * Unit test automation framework: Junit/TestNG * Programming language - Java |
| Exit Criteria | * Unit tests have passed successfully * Module is baselined in GitHub repo * Demo presented to BEIS PO & Supplier’s BA * All unresolved issues identified during this stage have been agreed, documented and handed over to next stage with appropriate and agreed resolution plan |
| Sign-Off/ Completion | * Although this testing is internal to the Supplier’s development team, BEIS will have complete visibility of the test progress and will be discussed in the daily scrums * Once features demoed are accepted by BEIS, they are approved and marked for deployment to the staging environment for further tests to be carried out by the Supplier’s test team |

## Acceptance testing (AT)

Acceptance tests are carried out by the supplier’s sprint test teams to determine whether the service meets the required user story specifications in an integrated environment viz. staging environment. The supplier’s sprint test team conducts these tests on two fronts:

* Portal frontend (GUI)
* Experience, process and system APIs

### Portal frontend – GUI

GUI testing checks if the screen controls like menus, buttons, icons, toolbar, menu bar, dialog boxes, fonts, colours etc. are delivered as per the approved UI wireframes and the functionality works to the specifications. The key focus of testing will be on the design structure, links, contents/images are all displayed correctly and do not shrink, crop or overlap when user resizes the screen. The key test objective should be the user does not get frustrated with the system interface.

The supplier’s test team should make use of GUI automation tools like Cucumber, selenium and automate using BDD test automation framework that allows tests to be executed at speed. Testing would cover:

**Positive Testing:** This validates the test functions correctly by entering a known value that is correct and verifying that the data/view return is in line with expectations.

**Negative Testing:** This validates that the test fails by entering a value that we know is incorrect and verifying that the component or test case fails.

### Experience API, Process and System APIs

The Transparency DB proposes the use of 3 types of API services:

* **Experience APIs:** that are tailor made APIs specific to BEIS consuming portals
* **Process APIs:** orchestrates and controls the data flow across business services
* **System APIs:** exposes records pertaining to system data

The supplier’s test team aims to check the functionality, reliability, performance and security of the various types of APIs mentioned above. The supplier’s test team would create the request payloads as per design, ensure requests are received and the responses are returning the correct information in a format that can be translated. Tests focusing on the service layers are dependent on the story in play.

The detailed test activities performed during this phase are illustrated in the table below:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Owner | * Supplier’s test team |
| Objective | * Validate that the solution meets the UI/UX requirements and are aligned to the wireframes and is considered functionally and technically ready for delivery into future test phases. |
| Entry Criteria | * Completed Unit Testing * UI/UX wireframes has been approved * Staging environment available with user ids and required authorisations * Solution configuration & build completed supplier testing start * Test data exists * Testing tools set up |
| Activities | * Each individual system configuration element will be tested * Overall functionality will be tested including browser compatibility testing and multi-platform testing for GUI * API testing of service endpoints |
| Environment | * Staging environment |
| Test data | * Mock up test data prepared by Supplier’s test team * Where possible automation will be used to create the test data |
| Tool | * IDE – Intelli J * BDD – Cucumber * Framework: Selenium/Serenity * Build – Maven * Repo – GitHub * Programming language – Java * API testing – Postman, RestAssured |
| Exit Criteria | * All planned tests are completed successfully * Test results have been reviewed and approved by BEIS PO and Supplier’s BA * All defects identified during this stage have been agreed, documented and handed over to next stage with appropriate and agreed resolution plan * Test results and reports produced |
| Deliverables | * Automated test cases * Defects log * Test reports |
| Sign-Off/ Completion | * Rather than written sign-off, formal Agreement between technical and functional leads will mark completion of the stage: based on the guiding principle that GUI and API tests has been successfully completed |

## Database testing:

Database testing shall be carried out by the supplier’s sprint test team to validate the metadata, key constraints, null able vs non-null able columns, datatypes etc. The key focus is to ensure tests are successful with the datatypes validated as per the data dictionary, logical and physical data models. As much as possible, tests are expected to the automated by the supplier’s test teams.

## Exploratory testing

In exploratory testing, both the test execution and the test design phase go together. This type of testing focuses on interacting with the application directly. Exploratory testing lets supplier’s testers “play with” the software that mimics possible user behaviours trying to find actions or edge cases that will break the system software. Testers do not document the exact process in which they tested the software, but when they find a defect, they document the test case that caused the failure. A round of light weight testing is usually attempted for the new feature delivered in the current sprint.

## End to End Testing (E2E)

E2E testing ensures all parts of the system and interdependent components (internal and external) operate in the desired fashion. It’s worth noting that the tests cover all interfaces and interactions between integrated components. Additionally test cases must be defined to test all functional requirements and expected user behaviours as specified by the Business. This testing is usually limited in comparison with functional acceptance tests and are carried out towards the end of the test phase but prior to the start of User Acceptance Testing. The E2E tests are written as part of the story and tests are expanded with new features accommodated as the sprints progress within the project. The E2E tests cover the full scope of the transparency DB.

|  |  |
| --- | --- |
| **Item** | **Description** |
| Owner | * Supplier’s test team |
| Objective | * To test the E2E user journeys and determine readiness before UAT |
| Entry Criteria | * All planned tests (Unit, UI, API, Exploratory etc.) should be completed * Integration requirements are completed and signed off * E2E design documents are completed and signed off * Staging environment is deployed with the latest codebase * Test resources are available |
| Activities | * Each integration level will be tested * **End to End Testing** will authenticate the flow of data to and from applications (Portals) associated with the application * **End to End flow** will be performed |
| Environment | * Staging environment |
| Test data | * Mock test data will be prepared by Supplier’s test team * Where possible automation will be used to create the test data |
| Tool | * Test Case Management Tool: Azure test plan * Test Task & Defect Management Tool: Azure DevOps |
| Exit Criteria | * All planned End to End test cases have been executed and results captured in Azure test plans * No critical and high Severity defects are open other than defects that have workaround and business approval to proceed for the next phase * All open medium severity defects have been reviewed with the stakeholders and were approved for deferral * Test cases, test scripts, test execution results updated in Azure test plan * BEIS PO & Supplier’s BA have been provided with test results |
| Deliverables | * Test results & defects |
| Sign-Off/ Completion | * Relevant BEIS PO/test representative will be responsible for sign-off |

## User Acceptance Testing (UAT / Private Beta)

User Acceptance tests or ‘Private Beta’ are formal tests executed manually by the BEIS PO or an UAT representative(s) to verify if the service satisfies its business requirements. At this point, the expectation is the entire application shall be up and running with a strong focus on replicating user behaviours. It’s worth noting that before UAT commences, the integration points and interfaces are already automated. UAT Tests will cover real-world usage of the system and will also cover off UX testing. Role-based scenarios will be used to ensure all aspects of the system are tested. It’s also worth noting that there should be a role dedicated to Supplier’s dev support throughout the entire duration of UAT. The users may also go further and measure the performance of the system and reject changes if certain goals are not met.

|  |  |
| --- | --- |
| **Item** | **Description** |
| Owner | * BEIS PO (Business Team will run the process) |
| Objective | * To ensure that the Solution delivered meets the business requirements |
| Entry Criteria | * Acceptance tests and all integration tests for GUI & API are complete as per the exit criteria * UAT test scenarios are written by BEIS PO / UAT representative(s) * Staging environment is available for conducting UAT tests * Software and approved versions of code installed * BEIS PO or UAT representative(s) identified for testing |
| Activities | * + All E2E user journey will be tested as per required roles and authorisations   + All agreed module specific business processes will be tested per required roles and authorisations   + Usability and customer experience testing of portals |
| Environment | * Staging environment |
| Test data | * Production like data loaded in the staging environment |
| Tool | * Test Task Management Tool: Azure test plan * Defect Management Tool: Azure DevOps |
| Exit Criteria | * All planned UAT test scenarios have been executed * Test scenarios not executed have been reviewed and approved with the business owners * No blocker, critical and high severity defects are open other than defects that have workaround and business approval to proceed for the next phase * All open medium severity defects have been reviewed with the stakeholders and were approved for deferral * Test cases, test scripts, test execution updated in Azure test plans * Service approved by BEIS PO / UAT representative(s) |
| Deliverables | * Test Results * Defects logs |
| Sign-Off/ Completion | * Relevant business stakeholders will be responsible for sign-off |

## Performance Testing

Performance testing will ensure that the system meets BEIS performance and throughput requirements. This stage of testing will also cover off load testing which will help to test the behaviour and performance under normal and anticipated peak load conditions.

The detailed test activities performed during this phase are illustrated in the table below:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Owner | * Supplier’s test team |
| Objective | * Ensure all parts of the system and all interdependent components operate in the desired fashion. |
| Entry Criteria | * Design documents have been approved by BEIS solution architects * UAT/Private Beta is complete * Staging environment is free from use by supplier’s functional testers * No outstanding defects (Blocker/Critical) * Non-functional requirements has been signed off |
| Activities | * Carry out tests against business processes or transactions using scenarios from previous testing phases |
| Environment | * Staging environment |
| Test data | * Performance Testing Team to define the test data based on the no of users required for this project |
| Tool | * Testing Tool: * Jmeter * Selenium |
| Exit Criteria | * 100% execution coverage of all planned performance test cases * Performance Test objectives are met * Performance Test report published & signed off * No Critical or High Errors remain open * Signed off Test Exit Report (TER) * There is no open Risk during PT closure, in case test objectives are not met |
| Deliverables | * PT schedule * PT cases * PT results * Defects log in JIRA * Performance Test Summary Report |
| Sign-Off/ Completion | * Relevant PO / business stakeholders will be responsible for sign-off |

## Security Testing

Security testing will help to evaluate the overall security of the “To Be” Solution by trying to exploit vulnerabilities that could be present within operating system and its related applications. The supplier’s security test champion will be placed who will be responsible for ensuring that the daily scans and security checks take place.

Vulnerability testing will also be carried out as part of Security testing. This will help to find and measure the severity of vulnerabilities within the “To Be” solution. The output of these vulnerability tests will be a list of vulnerabilities that have been identified which are prioritised by severity and criticality to the business.

*Note: Security – Penetration testing is expected to be carried out by BEIS security teams or a nominated BEIS 3rd party*

|  |  |
| --- | --- |
| **Item** | **Description** |
| Owner | * Supplier’s Test team |
| Objective | * Ensure all parts of the system and all interdependent components do not have any security risks at the time of the assessment and operate in the desired fashion. |
| Entry Criteria | * System design documents have been approved by BEIS prior to the development of the Security Test plan. * UAT/Private Beta is complete * No defects outstanding * Staging environment is available |
| Activities | * **Test Planning**: Create, publish and maintain a Security Test Plan for (Project) * **Test execution**: Security testing will be executed for the defined Security test cases |
| Environment | * Staging environment |
| Test data | * Supplier’s security tester to define the test data required for this project |
| Tool | * Vulnerability testing: OWASP Zed Attack Proxy (Proposed) |
| Exit Criteria | * Security Test objectives are met * Security Test report published & signed off * No Critical errors remain open * Publish Test Closure Report (TCR) * Remediation and mitigation plans accepted with BEIS stakeholders |
| Deliverables | * Security Test Plan (application testing only) * Test schedule * Test cases * Test results * Defects log in Azure DevOps * Security Test Summary Report |
| Sign-Off/ Completion | * BEIS security team and relevant stakeholders will be responsible for sign-off |

## Accessibility testing

Accessibility testing is a type of system test designed to determine whether individuals with disabilities will be able to use the system in question, which could be software, hardware, or some other type of system. Disabilities encompass a wide range of physical problems, including learning disabilities as well as difficulties with sight, hearing and movement. These tests are usually carried out against an industry standard document. A light accessibility test can be used derived from the regression pack. Some recommended tools that supplier’s test teams can use to carry out accessibility testing are JAWS and NVDA. Testing will be carried out as per WCAG 2.1 Level AA standard.

|  |  |
| --- | --- |
| **Item** | **Description** |
| Owner | * Supplier’s Test team |
| Objective | * Ensure that application is being usable for people having disabilities like hearing, colour blindness, old age and other disabilities. Accessibility standard is expected to be met as per WCAG 2.1 Level AA standard |
| Entry Criteria | * System design documents have been approved by BEIS * Staging environment is available |
| Activities | Testing the service with the most common assistive technologies and validate the new service against WCAG 2.1 Level AA standard |
| Environment | * Staging environment |
| Test data | * As needed |
| Tool |  |
| Exit Criteria | * Accessibility Guidelines are met * Accessibility test report published |
| Deliverables | * Accessibility test report * Test cases * Test results * Defects log in Azure DevOps |
| Sign-Off/ Completion | * BEIS PO and relevant stakeholders will be responsible for sign-off |

## Operational Acceptance Testing (OAT)

Transparency DB is setup as a new service within BEIS and hence ‘Operational readiness testing’ will be carried out as the final stage of testing once all other testing activities have been performed and the “To Be” solution is ready for live deployment.

OAT acts as a gatekeeper of the production environment, this will ensure that all the operational requirements pertaining to application implementations are tested thoroughly. This testing will act as the final quality checkpoint prior to deployment of the service into production. OAT testing can often be complex, with support needed from development, architects and infrastructure teams. Any constraints to conduct OAT will need to be detailed within the test plan around dedicated pre-production environment requirements and test execution hours (e.g. some tests may have to be done during non-business hours)

The detailed test activities performed during this phase are illustrated in the table below:

|  |  |
| --- | --- |
| **Item** | **Description** |
| Owner | * Supplier’s OAT tester |
| Objective | * Ensure all parts of the system and all interdependent components operate in the desired fashion. |
| Entry Criteria | * UAT/Private beta is complete * Integrated test environment is built and operational as per requirements * Software and versions of code installed * Base data set up is complete |
| Activities | OATwill be performed by Supplier’s OAT tester. Key validations as part of OAT may include (will be validated once NFT requirements are in place):   * NUT – Bandwidth/Latency * Server reboot testing * Monitoring/ Alerting/Logging * Resilience/Failover Testing * Backup/ Restore Verification * Deployment Rehearsal as per the release notes/deployment plan * Compatibility & Co-existence * Remote Connectivity * Archiving & Housekeeping verification * Cut-over day Simulation * Disaster Recovery Testing |
| Environment | * Staging / Pre-Production Environment |
| Test data | * Production like data is loaded in the test environment |
| Tool | To be discussed with BEIS operations teams |
| Exit Criteria | * Go-ahead for production deployment from Business and Operations * All planned OAT tests have been executed and results captured * No critical and high severity defects are open other than defects that have workaround and business approval to proceed for the next phase. * All open medium severity defects have been reviewed with the stakeholders and were approved for deferral * Test cases, test scripts, test execution results updated in * Test Exit Report is published to BEIS stakeholders/operations team |
| Deliverables | * OAT checklists * Test results * Defects log |
| Sign-Off/ Completion | * Relevant business stakeholders will be responsible for sign-off |

# Public Beta

Once the system has gone live, there will be a short phase of Public beta testing which will look to verify the release of the new system on the live environment. This phase will help to identify any issues surrounding the data, overall deployment or other impact areas that were not known / highlighted at the time of testing. Any issues found during this stage will be reported and communicated to the relevant business stakeholders who the issue could be affecting. If any critical issues are found then these will need to be communicated immediately as a decision will need to be taken on whether the issue will impact business critical processes. Wherever possible this stage of testing will be automated.

# Definition of Ready and Definition of Done

Below is the proposed DOR and DOD guide lines, however actual criteria to be selected by the team.

## Definition of Ready

Before the story is picked for sprint, all items listed below are achieved:

* Acceptance criteria
* Definition by example
* Test requirements
* Technical architecture
* Visual designs
* Unit/integration tests involved
* Negative path scenarios
* External Dependencies

## Definition of Done

Before the story is release for production, all items listed below are achieved

* All user story related tasks created by team for sprint are done
* Meets acceptance criteria given for user story
* API acceptance test coverage
* All test scenario/cases are successfully executed on SIT environment
* Defects found are raise: Defect life cycle
* Demo to PO/stakeholders done
* PO/Stakeholders are happy to sign-off the story

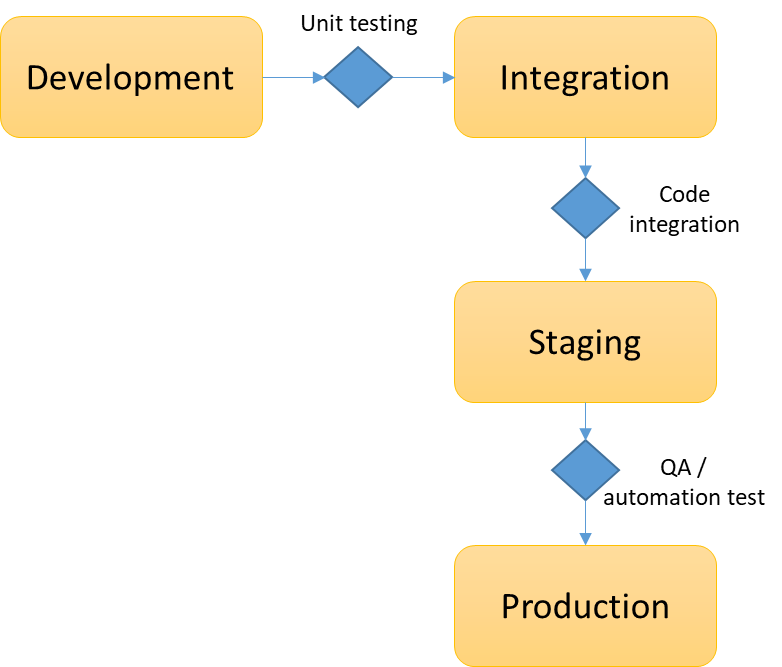
# QA and automation tools

Found below are the testing tools that are proposed to be used by Supplier’s testing teams as part of the testing process. Specific information on which tools will be used to test specific parts of the solution will be defined at a later stage.

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Test type** | **Tool** |
| 1 | Test management | Azure Testplan |
| 2 | UI automation | Selenium |
| 3 | API automation | RestAssured |
| 4 | Source code repository | GitHub |
| 5 | Integrated dev environment | IntelliJ IDEA |
| 6 | BDD | Cucumber |
| 7 | Test framework | TestNG/Serenity |
| 8 | Build automation tool | Maven |
| 9 | Service testing (Manual) | Postman |
| 10 | Database | PostGreSQL |
| 11 | Multibrowser & multiplatform | Browserstack |
| 12 | Performance test scripting | JMeter |
| 13 | Programming language | JAVA |

# Environments

At BEIS, preproduction environments are where the new services are built and tested. During the BETA phase, this approach to building test environments would be followed by the Supplier’s teams. The pre-production environments shall most likely include a: development, integration and a staging environment. The diagram below illustrates the high level gating process of the environments:



*Environment testing overview*

The requirements for each environment type are defined in the subsections below. It is worth mentioning that there may be multiple instances of each environment, e.g. multiple development environments, or multiple test environments.

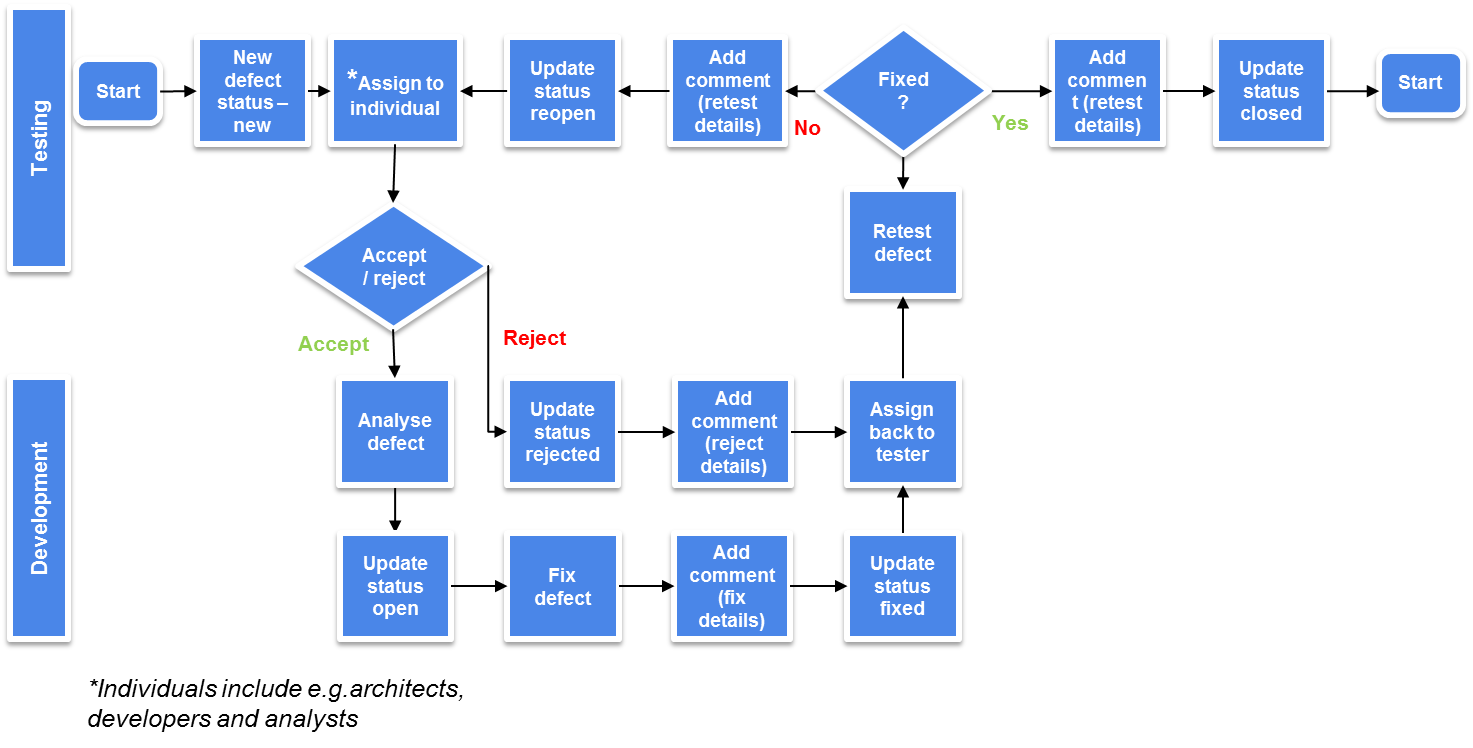
The promotion criteria mentioned in the following sections refers only to the technical architecture testing; the comprehensive promotion criteria are defined within the SDT End to End Testing Strategy document.

|  |  |
| --- | --- |
| **Environments** | **Summary** |
| Development | This environment allows developers to perform unit testing on their newly implemented code, before promoting code changes to testing. |
| Integration | This environment allows the build team to combine all code and verify if it works as intended |
| Staging | Staging provides a true basis for Quality Assurance testing, because it precisely reproduces what is in production, and areas such as private beta, performance etc. can be tested properly. |
| Production | This environment is where public beta testing is performed after an initial verification of the environment following code deployment. |

# Defect Management

During the course of testing, both defects and enhancements will be identified and logged in Azure DevOps. A defect is subject to the change control process in that it requires either a fix or a change to the initial system set-up. In general, a defect can be defined as an item being non-compliant with requirements. An enhancement, on the other hand, can be defined as a change suggesting a different or additional functional behaviour is desired, which is not specified in the requirements and is therefore subject to change control. Furthermore, defects can be classified by their severity.

Each supplier’s tester will be responsible for logging defects. The Supplier’s test manager will track and report all defects in for all phases of QA testing. This will allow for centralized tracking of defects, standardized defect tracking, and easy reporting. An indicative example how the defect process will work can be found in the image below.



*Detailed view of sprint process showing split between design, dev and execution phase*

**Defect Status:** The below table depicts different status in defect life cycle.

|  |  |
| --- | --- |
| **Defect Status** | **Description** |
| New | The default status for a newly created defect |
| Assigned | Defect has been reviewed and assigned to relevant team |
| Open | Defect has been accepted and defect fix is work in progress |
| Fixed | Defect has been resolved, Code changes ready for promoting to Testing environment |
| Reopen | Defect did not pass in re-test closure |
| Reject | Defect that has been rejected and not accepted |
| Accept | Defect has been accepted |
| Closed | Issue Resolved |

**Defect Severity and Priority**

When a defect is raised, an individual severity and priority rating is assigned. The severity will indicate how critical the defect is to the system. Where the priority is used to prioritize the defects, for example, should there be several severity 1s then the highest priority defect would be investigated first.

|  |  |  |
| --- | --- | --- |
|  | **Perspective** | **Definition** |
| Severity | IT | Used to describe the gravity of an undesirable occurrence – that is, the effect a defect has on the solution in terms of its impact |
| Priority | Business | Used to compare two things where one has more importance – that is, the urgency with which a defect is required to be resolved |

**Defect Severity**

|  |  |
| --- | --- |
| **Severity** | **Description** |
| Blocker | Cannot proceed with testing. The defect affects critical functionality or critical data. It does not have a workaround.   * Cannot login * Cannot upload data |
| Critical | Sever defect, which could stop part or the majority of testing. Therefore requires immediate investigation and resolution. The following points could also contribute to a critical failure:   * Data loss or data corruption * Critical functionality discrepancy between requirements and implementation * Blocking issues which prevents testing activities to continue * Data or display inconsistencies (web and/or paper based) * Security concerns with customer visibility * Severe performance degradation or instability of the system * Security concern with customer impact   Critical functionality error handling failures with high occurrence and resulting in high impact to customer usability |
| High | Severe defect but does not stop testing. Some of the functionality is unstable, but can progress with testing up to a point. Therefore requires prompt investigation and resolution.   * Critical functionality impairment with less than a desirable work around (e.g. manual process) * Failure of non-critical functionality with no work around * Non critical functionality discrepancy between requirements and implementation * High usability (UI) discrepancies * Security concern with no customer impact * Incorrect / invalid configuration data |
| Medium | Although unstable, everything can be tested, perhaps with work around, but would have a business impact. Requires resolution before next cycle of testing.   * Non critical functionality impairment with an acceptable work around * Non critical functionality error handling failure * High usability cosmetic issues such as key spelling errors which could create user confusion * The problem occurs rarely and can quickly be recovered by operator intervention |
| Low | A defect that requires amendment prior to going live but does not require immediate investigation.   * Rare occurrence – the issue is not readily reproducible * Low customer impact feature failure * Cosmetic issues not easily recognizable (e.g. low impact spelling and colour schemes) * The defect is in the test procedure and not in the system itself |

### Defect Priority:

The table below highlights a summary of the different defect priority levels. It will be important to ensure that these priorities are aligned to the sprint methodology that has been mentioned above.

|  |  |
| --- | --- |
| **Priority** | **Description** |
| Critical | Critical, resolution required ASAP, ideally within 24 hours   * Blocks further testing or initiative progress * Currently very visible and/or detrimental to users * Needed for time critical deadline * Possibly immediately detrimental to revenue or Inmarsat reputation |
| High | High, resolution required before the next release into testing   * Potential for numerous user complaints to be raised * Critical area of the solution   Will be detrimental when released   * Does not conform to what was stated as a requirement for the solution |
| Medium | Medium, resolution required before final solution go live   * Not a critical area of the solution * Some users may be impacted but a work around is known * Very few user complaints likely to be raised |
| Low | Low, resolution if time permits but could be moved to future release   * Would like to fix but can be released as is * Few users likely to even notice much less likely to be impacted or raise complaints |

# Status Reporting & Meetings

Each testing phase will have the ability to create its own meeting schedule, however, the schedule proposed below is considered as a standard approach to adopt for the BEIS transparency DB project.

|  |  |  |
| --- | --- | --- |
| **Meeting** | **Description** | **Frequency** |
| Daily Sprint Scrums | Update of testing activities that have occurred and an overview of issues and resolutions. (BEIS PO, Supplier’s sprint team) | Daily |
| Weekly governance review | Update of testing activities that have occurred, an overview of issues and resolutions, and planning for any updates required by the system. (BEIS PO, Supplier’s delivery manager with inputs from Testing Manager) | Weekly |

# Testing Governance

Found below is a high-level approach to how the testing process will be governed on the SDT Programme. A separate more detailed document will be produced to explore each point listed below but at a minimum, the following test governance points will be followed:

**Planning** – the test plan will be outlined in a separate document that this project may implement and operate. The purpose of this plan is to establish and maintain the milestones and testing activities that need to be delivered in order to achieve the project objectives

**Meetings** – during the testing sprints there will be a range of daily and weekly meetings held to ensure that everyone is aligned on the testing activities that are due to take place.

**Reporting –** throughout the testing sprints the testing work stream will be expected to feed into the Supplier’s weekly governance report, which highlights the key tasks, risks and issues that need to be highlighted for the reporting period in question.

**Management of risks, actions, issues and dependencies -** There is a RAID Log in usage as part of the weekly governance meeting with BEIS. Risks, issues and dependencies relating to testing will be stored within this document and it is expected that these will be updated on a regular basis.

# Suspension Criteria

The BEIS Project Manager in conjunction with BEIS PO, the Supplier’s Delivery manager and the Supplier’s test manager will determine if testing should be suspended. The suspension criteria for the test cycle is a condition found that is severe enough to preclude any additional system testing from being executed, or a functional change in the design or configuration which would invalidate any continued testing. The product may undergo retesting and/or additional testing to demonstrate the successful remediation of the issues prior to the resumption of QA testing. QA testing will commence again once the resumption criteria have been met.

Few more points for suspension criteria is listed below:

* Build not fit for testing
* Environment issues i.e. unavailability of environment for testing
* Lack of clarity on requirements, if any
* Show stopper defect
* Budget constraints
* Build does not include all functionalities agreed in scope
* Unavailability of reference and other test data
* Late deployment of build impacting the sprint timelines

# Resumption Criteria

The resumption criteria may be, but is not limited to:

* A valid work-around which can be employed and not cause any questioning or uncertainty of test results achieved during the test execution.
* A final resolution to the incident which triggered the suspension of system testing.
* Environment becomes available for testing
* Clarity on requirements
* Show stopper defects closed
* Budget constraints overcome
* Programme team may override the decision to suspend testing.
* Unavailability of reference and other test data
* Late deployment of build impacting the sprint timelines

# Risks Management

Defined testing risks can be found in the Project Risk Register, where all risks for the SDT Programme can be found. The following risks will be monitored and tracked during the various testing stages.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk** | **Likelihood Of Risk** | **Impact Of Risk** | **Mitigation Strategy** | **Owner** |
| Technical development may not be completed in time to meet sprint testing dates | Medium | Medium | · Complete technical design specs as early in the process as possible  · Involve right stakeholders to review all technical specs | Supplier’s technical team |
| System errors and resulting patch application may prevent testing and delay the test schedule | Medium | Medium | Test as much of the system functionality as possible in early stages of testing (i.e., Unit) to identify system issues | Supplier’s technical and functional test teams |
| Since the “To Be” solution will be a completely new service, there may be more system bugs and patch releases than usual | Medium | Medium | · Review patch release documentation and perform automation testing regularly as needed | Supplier’s technical team and QA teams |
| No test data in the system | High | High | Import test data into the system before testing | Developers / Test Team |

# Escalation Procedures

During the various testing phases, time can be of the essence in issue resolution. Once an issue has been identified by anyone involved in the testing process, it should be reviewed with the Supplier’s test manager. Upon review, the tester should enter the issue into Azure DevOps with as much detail as possible, including screen shots if applicable. The tester should assign the issue to the appropriate development resource and include the issue severity. The supplier’s test manager will be responsible for tracking and working the open issues. Specifically, the test manager has the following responsibilities:

* Report all new issues during daily test status meetings
* Escalate issues as required >Supplier’s Delivery Manager > BEIS PO> BEIS PM
* Follow-up on open issues daily until closed

# Quality Gates

Quality control Gates are part of End-to-End quality that are periodically measured and monitored by programme management team to evaluate quality of work, right from project initialization until the end of the project life cycle. Quality gates will be run at the end of each test sprint.

The Gate criteria tabulated below will be established after a concurrence with the IT and the business testing teams to govern the release of code and software components to the next testing stage. The percentages for test execution and pass rates are outlined in the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. **Quality Gate** | 1. **Testing Stage** | 1. **Next Stage** | 1. **Criteria to move for Current Stage to Next Stage** | |
| 1. **Test Execution** | 1. **Defects** |
| 1. **QG-1** | 1. **Unit Testing** | 1. **Acceptance Testing (GUI & API), exploratory, database testing & E2E testing** | 1. % Unit Test Cases(TDD) Attempted = **100%** 2. % Unit Test cases (TDD) Completed successfully = 95% | 1. Zero critical and high severity open Defects |
| 1. **QG-2** | 1. **Acceptance Testing (GUI & API), exploratory, database testing, E2E testing** | 1. **UAT/Private beta** | 1. % Test Cases Attempted = **100%** 2. % of Test Cases Completed successfully = **95%** | 1. No critical open defects and high Severity defects are open other than defects that have workaround and business approval to proceed for the next phase. |
| 1. **QG-3** | 1. **UAT/Private beta** | 1. **Performance testing, Accessibility testing** | 1. % of Test cases Attempted = **100%** 2. % of Test cases (cycles) Completed successfully = **95%** | 1. No critical open defects and high Severity defects are open other than defects that have workaround and business approval to proceed for the next phase. |
| 1. **QG-4** | 1. **Performance & Accessibility testing** | 1. **Security Testing (application only)** | 1. % of Test cases Attempted = **100%** 2. % of Test cases (cycles) Completed successfully = **95%** | 1. No critical open defects and high Severity defects are open other than defects that have workaround and business approval to proceed for the next phase. |
| 1. **QG-5** | 1. **Security Testing** | 1. **OAT** | 1. % of Test cases Attempted = **100%** 2. % of Test cases (cycles) Completed successfully = **95%** | 1. No critical open defects and high Severity defects are open other than defects that have workaround and business approval to proceed for the next phase. |
| 1. **QG-6** | 1. **OAT** | 1. **Go Live testing** | 1. OAT checklist attempted = **100%** 2. % OAT checklist tests completed successfully = **95%** | 1. No critical open defects and high Severity defects are open other than defects that have workaround and business approval to proceed for the next phase |

# Skills Required

Several different types of skills may be required throughout testing depending upon the testing resources’ role in each phase. The table below outlines the skills that will be needed, as part of the testing process.

|  |  |
| --- | --- |
| **Training** | **Description** |
| Script development | Testers to understand the basics of how to create edit record and publish scripts. |
| Scripting Overview | Testers to understand how to run and document their testing results to increase the initial script pass rate and ensure correct recording of all scripts. |
| Solution Navigation | Any new testers should have a basic understanding of the “To Be” screens and its navigation. |
| Knowledge of use of QA tools for various tests stages | Depending on the testing approach, training will most likely be required on the tool prior to the Performance Testing phase. |
| Knowledge of Network/Database monitoring tool(s) | If no internal resources can maintain, set-up and/or run the monitoring tool(s), then additional training is required prior to the Integration phase. |

# Appendix (To be updated)

## Acronyms and Abbreviations

The following tables list acronyms, abbreviations and terms used in this document.

|  |  |
| --- | --- |
| **Acronym** | **Term** |
|  |  |
|  |  |
|  |  |
|  |  |

## Terms

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Test Plan | A document describing the scope, approach, resources, and schedule of intended test activities for a system or related set of systems development. Identifies test items, features to be tested, summary of testing tasks, who will do each task, and any risks requiring contingency planning. |
| Test Case | A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement. |
| Test Script | Documentation specifying inputs, predicted results, and a set of execution conditions for a test item. Test scripts are used in automated testing. |

1. Not applicable for Public facing website [↑](#footnote-ref-1)