PPL MASTER TEST STRATEGY

v1.9 – 21st July 2016

INDEX

1 DOCUMENT CONTROL

1.1 History

Version	Date	Change Description	Changed By
1.0	06/10/2015	Initial draft	Dan Turner
1.1	13/10/2015	Revised based upon initial feedback	Dan Turner
1.2	30/10/2015	Final draft for wider audience review	Dan Turner
1.3	11/11/2015	Incorporation of review comments from wider audience review	Dan Turner
1.4	16/11/2015	Project team feedback incorporated	Dan Turner
1.5	17/11/2015	Terrorism COB defined	Dan Turner
1.6	19/11/2015	Feedback from Ebix incorporated	Dan Turner
1.7	11/01/2016	Revised in line with feedback received from MAT strategy review	Dan Turner
1.8	30/03/2016	Updated references to PPL	Dan Turner
1.9	21/07/2016	Revised for FinPro release	Dan Turner

2 GLOSSARY

Acronym	Explanation
API	Application Programming Interface
BEV	Bird's Eye View
BRD	Business Requirements Document
CR	Change Request
E2E	End-to End
DB	Database
CI	Continuous Integration
FAC	Facultative business
FAT	Field Acceptance Test
FINPRO	Financial and Professional lines
LIIBA	London and International Insurance Brokers' Association
IUA	International Underwriting Association
K&R	Kidnap and Ransom
LMG	London Market Group
LMA	Lloyd's Market Association
MAT	Market Acceptance Test
PPL	Placing Platform Limited
PM	Project Manager
SA	Solutions Architect
SME	Subject Matter Expert
TM	Test Manager
UI	User Interface
UAT	User Acceptance Test
URL	Uniform Resource Locator (website address)

3 INTRODUCTION

3.1 Executive Summary

3.1.1 Audience

As this is the master test strategy, it is intended for review by the PPL project team, vendor and market PMs. This is the master test strategy to capture all phases and methodologies and, accordingly, has a technical level of detail.

An individual strategy tailored for the MAT test phase will be produced and distributed to the relevant audience separately.

3.1.2 Approach

This is the master test strategy for the PPL implementation (please see the 'Overview / What is PPL?' sections immediately below for an explanation of PPL).

Essentially, there are three main cycles of testing for PPL:

- Development and system test – owned and performed by Ebix
- Functional system test – owned and performed by PPL
- MAT – owned by PPL and executed by carriers and brokers

It covers the full test lifecycle, through:

- Development testing by vendor
- System testing by PPL
- Market Acceptance Testing (MAT) by carriers and brokers
- Non-functional Testing (performance and security)
- Go-live tests

This strategy document is organised into distinct sections, describing:

- Scope (both in-and-out)
- Test phases that comprise the PPL test effort
- Documentation relevant to the testing of the implementation
- Test methodologies employed
- Details of the design of the test plan (test plan being defined as a schedule in this context)
- Test data to be used in each test phase
- Assumptions
- Risks and mitigation
- Communication plan

It is intended to provide the full detail of how PPL testing will be implemented, what it entails, and by whom.

The platform will be developed and initially tested by the vendor, Ebix. Upon receiving confirmation that the code has been internally tested by Ebix (including the publishing of test reports and sharing test scripts), it will be made available to the PPL test team for them to conduct the system test activity (a technical test to ensure that the platform meets its functional requirements as specified, using positive and negative testing techniques).

Following confirmation of a successful PPL system test, the MAT phase of testing will be commenced, which will require a co-ordinated effort from brokers and carriers to test the various in-scope business processes. In parallel with MAT, non-functional testing will be executed.

Completion of MAT and non-functional testing will lead to the transition into go-live.

3.2 Purpose of document

The purpose of this document is to detail the test strategy for the PPL platform and the collaboration between the component workstreams that comprise the project:

- PPL
- Ebix
- LMG
- LMA
- LIIBA
- IUA
- Brokers and Carriers

3.3 Involved Parties

The associations will provide details of test resource but will not be involved in the day-to-end testing. The Associations (LMG, LMA, IUA and LIIBA) work in partnership in the interests of their members.

- Placing Platform Limited (PPL) team
    - PPL are the delivery team for the PPL application. PPL will author the business requirements and functional documentation required prior to development of the platform.
- London Market Group (LMG)
    - Senior market-wide body with a primary function to act as the champion of the modernisation agenda in the London market. LMG will provide the assurance that of PPL has been delivered according to the business requirements. LMG will oversee the project management, testing, and delivery to go-live.
- Ebix
    - Ebix are the vendors. They must provide the software solution, the evidence that it has been tested prior to handover to the PPL test phases (including test scripts), and ongoing support during the test phases.
- Lloyd's Market Association (LMA)
    - The Lloyd's Market Association (LMA) represents the interests of the Lloyd's community, providing professional and technical support to their members. All managing and members' agents at Lloyd's are full members. Through the LMA, their interests are represented wherever decisions need to be made that affect the market.
    - The purpose of the LMA is to identify and resolve issues that are of interest to the Lloyd's market. They work in partnership with the Corporation of Lloyd's and other market-related associations.
- London and International Insurance Brokers Assocation (LIIBA)
    - LIIBA represents the interests of Lloyd's insurance and reinsurance brokers operating in the London and international markets.

- IUA
    - The International Underwriting Association of London. The IUA represents the interests of member companies that are fully authorised international wholesale insurance and reinsurance businesses operating in, or through, London. Through the IUA, their interests are represented wherever decisions need to be made that affect the market.
    - The IUA is involved with of PPL to ensure that it meets the business requirements of their members, and they will be involved in the MAT phases of testing by ensuring IUA members are engaged by the PPL Test Manager.
- Brokers and Carriers
    - Representatives from the brokers and carriers community who will be involved in the MAT testing.

3.4 PPL Overview

The PPL project is intended to deploy the following component parts of functionality to provide an integrated end-to-end solution:

- Quote
- Firm Order
- Endorsements

3.5 What is PPL?

The PPL Platform is a harmonised platform covering both Ebix MarketPlace system and LimeStreet systems. MarketPlace is currently used by Aon and its carrier partners. Limestreet is used by a number of market brokers and carriers including JLT. However, a number of elements have been requested to be enhanced by the market.

At a high level, the Placing Platform is a common technology-based service that will provide:

- Ability for brokers to submit and carriers to receive, and both to action: quote, firm order and endorsements;
- Development of defined standard integration capability to integrate with back office processes (subject to member specific technology integration capabilities), and with relevant core market technology components – this will be a future development;
- Ability for brokers and carriers to message between the platform and their own systems;
- Support for structured data capture for carriers;
- Provision of a common work management and tracking tool;
- An audit trail of actions undertaken on / via the platform such as holding the detail of the different versions of the placement record;
- A clear set of processes and standards for initiating insurance transactions to the London Market.
- Document comparison
- Endorsements functionality

4 SCOPE

4.1 In-Scope

The scope of PPL comprises:

- Quote
- Firm Order
- Endorsements
- The Dashboard

This ensures that transactions are capable of completing a full E2E path through the system.

PPL will provide the functional documentation to Ebix, prior to development of the PPL system test, of the requirements that the platform has been built against. These will drive the test effort.

4.1.1 Non-Functional Requirements

Non-functional testing will be a significant aspect of the test effort. The list of non-functional requirements is embedded below:

The types of testing that will be in-scope for the non-functional requirements are as follows:

- SQL injections (to show that SQL injections get caught)
- Load testing
- Stress testing
- Service denial
- Performance testing
- Security testing

For the definitions of each discipline, please refer to the Non-functional test section of this document.

4.1.2 Test Phases in Scope

Ebix Internal Dev Test

PPL System Test

MAT

Non-Functional Testing

Non-functional testing will run in parallel with the MAT test window.

Field Acceptance Test – Go-Live

4.1.3 Manual Testing

The testing conducted by Ebix, PPL and brokers and carriers post-unit testing will be exclusively manual tests. It is anticipated that the manual tests conducted by the PPL test team will be automated to form a regression test suite.

4.1.4 Collaborative testing

Development test capability will be provided by Ebix. No system testing will commence until Ebix can show that development testing has been successfully completed and that the component parts of the system integrate.

To assist in the timescales involved during system testing, Ebix will provide a limited system test facility.

Ebix will provide PPL with its test scripts used during system testing and a test report.

4.1.5 Quality Gates

The testing will be part of the Quality Gate methodology, where each test phase must have met its exit criteria before the subsequent next test phase can commence. This approach ensures that test phases commence with significantly less risk than if there was no / little evidence that the preceding test phases had completed successfully.

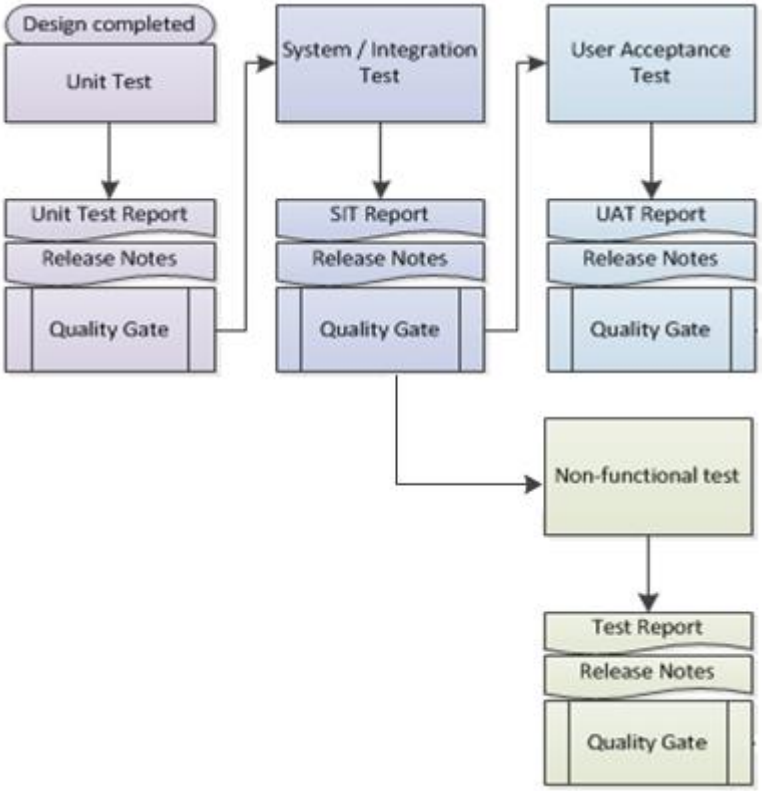

Figure 1 - Example Quality Gate structure

It is important to note the three main cycles of testing / responsibilities that are incorporated into the test effort:

- Ebix development and test, handing over to…
- PPL test team functional system test, handing over to…
- MAT conducted by brokers and carriers.

A non-functional test activity will occur at the same time as MAT, but this will be the responsibility of PPL of both PPL and Ebix.

4.2 Out of-scope

The following aspects are out of the scope of this test activity:

4.2.1 ACORD Messages

The ACORD messages require testing to confirm accuracy. Ebix currently has a test environment where it can test this. This is purely an Ebix responsibility.

The permutations of the messages supported are:

- Upload contract to Firm Order (ACORD Messaging channel)
- Download SLA completed placement (ACORD messaging channel)
- Upload Endorsement (ACORD messaging channel)
- Upload contract to Quote (ACORD messaging channel)

- Full endorsement cycle upload/download (ACORD messaging channel via TMEL)
- Request download (ACORD messaging channel)
- Response upload (ACORD messaging channel)
- Carrier SLA completed placement download (ACORD messaging channel)
- Firm Order “reconciliation message” download (ACORD messaging channel)
- Carrier completed Endorsement “reconciliation message” download (ACORD messaging channel)

4.2.2 TMEL Testing

This is a test of the messaging gateway between PPL and existing third-party platforms. This is out of scope for PPL tesing and is conducted jointly between Ebix and the TMEL team of Lloyd’s.

4.2.3 Automated testing

Automated testing is out of scope for the test framework of , other than as a regression suite for system tests. This is due to time constraints and the inevitable significant overhead in getting the manual tests converted to automated.

Post-, the core tests of the system will be automated to provide a regression suite.

No automation of the MAT tests will be in-scope for or any other phase. The automation testing will be limited to automating manual system tests to create a regression test suite for test coverage of any code changes or future releases.

It is important to note that the performance aspect of the non-functional testing will be automated in the majority. This is separate to the idea of automating functional tests, and is required to simulate voluminous loads of usage out of reach of manual input.

4.2.4 Model Office testing

There will be no model office testing for PPL. A model office facility will exist, but this will be purely a demo facility and not bound by acceptance criteria.

The MAT phase will mirror the MOT capability.

4.2.5 Go-live Configuration Testing

Ebix are required to test the new users’ configuration settings to ensure that the permissions / views they have requested are in place.

However, as part of the FAT activity, PPL will conduct an equivalence partition test of go-live users to provide assurance that configuration has been accurately set up.

4.2.6 Test phases executed in parallel (other than non-functional)

No test phase, with the exception of Non-functional, will run in parallel with each other.

For example, if system test and MAT run in parallel, and a significant defect is found in system test, the likelihood is that whatever has been tested to date in MAT has now been invalidated (due to the inevitable code change) and will have to be retested once the code fix has been deployed. It is not good practice to run test phases in parallel.

Non-functional testing can be run in parallel with MAT, typically because this is the first opportunity when the code is stable and won’t be significantly amended.

5 DOCUMENTATION

The correct documentation is critical to the test effort of of PPL.

5.1 Overview

Adequate documentation is critical to any test framework. It provides the foundation of the requirements of what the system must look like, through recording the test content, and including reporting the test outcome. Without adequate documentation in place, there is always doubt as to what the system has been developed against. Correct documentation facilitates the auditability of the whole test activity.

It is essential that development is only conducted against signed-off or approved documentation – not only to ensure that superfluous tests are avoided and defects unnecessarily raised, but also that the system is built to the specification and requirements requested.

The following artefacts must be available to the developers and testers:

- Functional Specification
- Business Requirements Document (BRD), as signed-off by all stakeholder carriers
- Technical Design Document (TDD)
- Test Strategy (both the master test strategy and the individual strategies per test phase)
- Test Plan (both the master plan and the individual plan per test phase), as signed-off by all stakeholder carriers
- Any other supporting documentation (such as screen mock-ups) must be formally signed-off by the business
- Release notes to show what is in each code-drop
- User guides or training manuals

The functional specification describes the requested behaviour of the required system.

The BRD contains the business requirements.

The TDD is used to specify the technical parameters of the system.

The test strategy sets out the vision of what will be tested and how it will be conducted. Although typically an evolving document, it must be signed-off prior to testing commencing.

- Scope
- Entrance / Exit Criteria per test phase
- Resources per test phase
- Test methodologuies involved at each test phase
- Risks and mitigation
- Defect Management

The test plan shows what is being tested, by whom, and when.

Supporting documentation would include screen design mock-ups and calculation formulae, for example.

6 TEST METHODOLOGY

6.1 Waterfall or Agile?

has been developed as a compromise between Agile and Waterfall methodologies. As an example, when the various CRs have been submitted and developed, functional documentation for each one has not been authored in some cases. This was a result of timescales and the need to get a system operational as a priority.

However, this has a negative impact on the test effort for the following reasons:

- Lack of functional documentation with which to understand the process flow of the system
- Not enough detail in the supplied documentation with which to construct detailed test requirements and test scripts – specifically for system testing, when the testing will be focussed on functionality other than the happy path (collection of simplest E2E processes).

For PPL the testing will, by default, be conducted on more of a Waterfall approach – due to the fact that the system has largely already been developed as test requirements are being authored. The testing will be reactive – in that a system will be provided with accompanying functional documentation and the test team will then author their tests and execute them. There is no provision within PPL for Agile testing, with functionality tested per sprint against a product backlog, for example.

6.2 Test Framework

The testing of PPL will conform to an industry-standard test framework.

A robust test framework can be considered as an iterative defect discovery process:

- Discovery of defects, failures and flaws
- Assessing the design limits of the system
- Testing the behaviour of the system
- Checking that the system does what is expected
- Checking that the system does not do what is not expected
- Showing that the system performs to user, design and development team satisfaction

It is important to note that testing is not simply the discovery and removal of defects – it is the process that assures users that the quality is correct.

The software development standard ISBN 0-7381-1559-2 specifies assurance as being the process of producing a guarantee as to the total quality of a product.

Software testing can be visualised in the following quality control and assurance process:

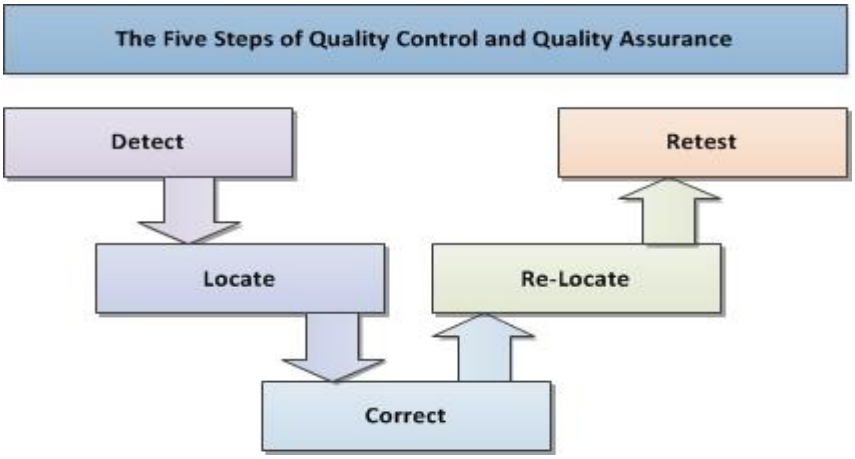

Figure 2 - Example of test framework quality overview

Any test framework must comply with industry-recognised standards of testing. This is to ensure uniformity of process and best practice.

BS 7925 – Software Component Testing:

- Plan the test (entry and exit criteria)
- Specify the test
- Execute the test
- Record the results
- Check for completion (assess and analyse the results)

IEEE 829. This directive ensures that the following documents are included in the test framework, which naturally structures the framework into a logical process flow:

- Test Strategy – describing what will be tested, and the methods planned to do so
- Test Plan – timescale view of what activities will be run at which point in time
- Test Requirements – linked to source documentation
- Test Scripts: including the design, required test data, action required, expected result, and success criteria
- Anomaly Report (or Defect Log)
- Test Reports – for each test phase

6.2.1 Test Governance

The repository where test requirements and test scripts are stored (and executed) is a central point to the test framework. This provides the levels of governance as to how the test process is run and reported on. Test requirements and defect tracking will be done on Google Sheets in order to ensure that the results can be shared with the vendor and interested parties in real time, as opposed to emailing documentation around.

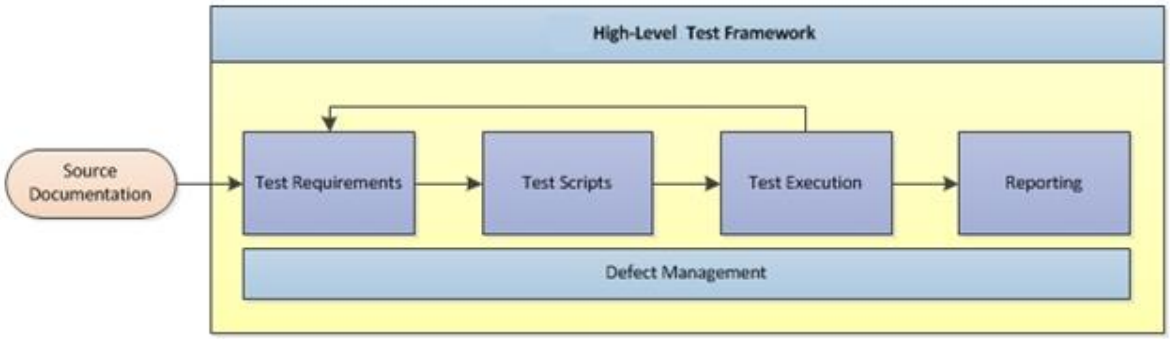

Figure 3 – Visualisation of high-level test framework

Requirements must be appropriate to the relevant phase of testing. For example, a system test would require test requirements to expand upon the functional specification and included boundary and negative tests – MAT requirements have to have more of a focus on business processes required as specified in the BRDs.

All test requirements must reference source documentation – whether that be functional specification, BRD, defect ticket, or change request. This is to ensure that the delivered system is as per the requested one.

Test Requirements must be grouped in their relevant repositories by functional area or by the grouping of the relevant items in the source documentation. This assists navigation and provides a logical structure.

6.2.2 Test Environments

It is important to note that test environments in the context of the PPL implementation refer to the location that the website is hosted on. The platform will be accessed via browser.

There is no requirement for carriers and brokers to create separate test environments for the MAT phase. A URL will be provided which will give access to the platform on whatever environment it is being hosted on.

Each test phase will be conducted on a separate test environment. There are several reasons for this:

- Test phases at the beginning of the test cycle (such as development and system test) require more flexibility and are therefore often in a state of flux / accepting regular code changes
- Once business users become involved in the testing, the environment must be close to life-like
- Any fixes found during the latter stages of testing will require testing in the unit and system test environments first – it is not good practice to deploy an untested fix directly into an MAT environment

For PPL, there will be no integration with any existing applications – all users will access the platform via the supplied url link.

- eeTest – this is the Ebix system test environment, for which they provide a system test capability to show that the development testing has produced the required outcome
- eeUAT – this is the environment PPL use for system test and MAT

6.2.3 Test Phases

As covered in the Scope section, the following test phases will be executed for :

- Static Testing (PPL and Ebix)
- Development testing (Ebix)
- System testing (PPL and Ebix)
- MAT (PPL)
- FAT (PPL)

In the example below, Model Office testing will not be in-scope for .

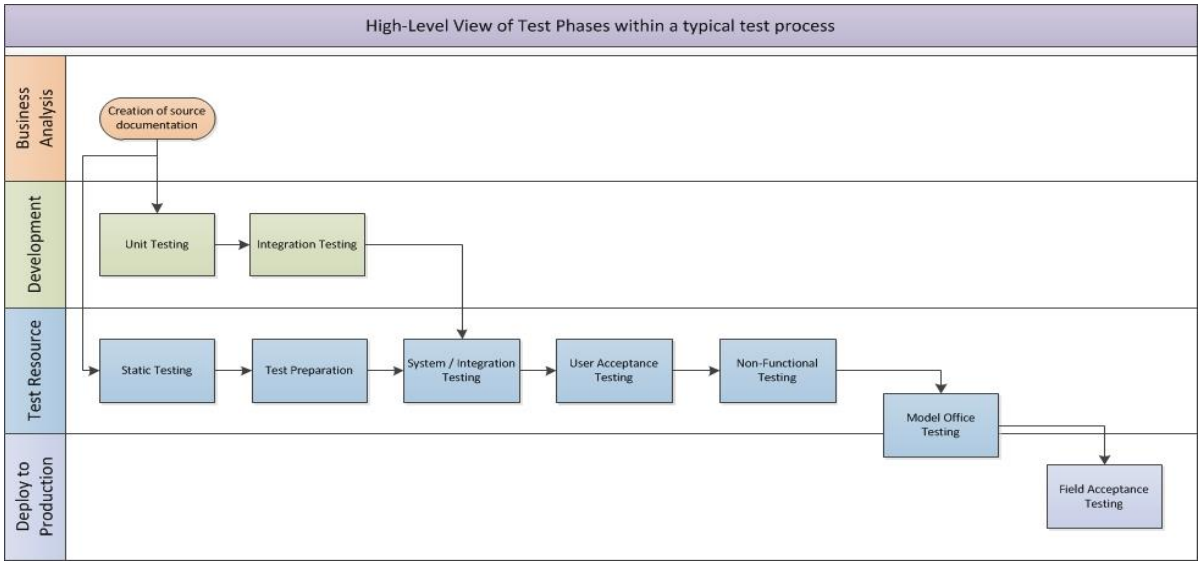

Figure 4 - Example of a high-level view of typical test phases

6.2.3.1 Development Testing

Ebix are responsible for providing the development testing.

Development testing must occur to show that the code, at its most granular testable level, executes and integrates – ideally the tests are triggered upon each code check-in. This will highlight immediately where any defects / code conflicts are and when they were introduced.

Because the overall PPL solution is comprised of multiple individual components, integration testing is paramount.

This is an Ebix responsibility, as it is the development resource who would perform tests to show that the components integrated. This must be done prior to handover to system test.

It is expected that the integration testing will be run along the lines of the Big Bang approach, whereby most of the developed modules are grouped together to form either an E2E software system or at least a significant part of the system.

The Big Bang approach has the advantage of time efficiencies in the integration testing process.

When code is checked-in, it must be accompanied with coders' comments to explain why the code has been amended (such as a reference to a defect ID).

All code must be source-controlled. Any change to the code must cause a new version of the code to be created, and the previous one to be kept as it was.

Code cannot be amended and deployed on the fly – organised code-drops must be used.

A Test Report is not required at the completion of development testing

Entry Criteria

- Documentation required for development (such as functional specification) produced and signed-off

Exit Criteria

- All planned components execute and integrate as expected

6.2.3.2 System Testing

Although labelled 'system test', this test phase also includes a degree of integration testing as a natural consequence of running transactions through the E2E system.

System test is the process of exercising the system to the prescribed limits (and beyond) of its documented boundaries to actively look for defects / inconsistencies. The purpose of this is to provide assurance that MAT can focus on coverage of business processes without fear of encountering functional defects. Testing methodologies expected (but not limited to) are:

- Positive testing
- Negative testing
- Boundary testing
- Error handling

The system test scripts have to be referenced to test requirements, which in turn reference source documentation. The functional specification would be the basis of this, intertwined with knowledge of the business processes gleaned from the BRDs. Standalone tests cannot exist – each one must be linked to a source artefact.

System test requirements and scripts must be prioritised according to what the business dictate as critical functional areas. The purpose of this is that it is very unlikely that all tests can be executed in the given timescales – therefore,

there must be a mechanism within the test framework that dictates what tests are run first, or indeed sacrificed if there isn't enough time to run all the tests.

System test must be run in complete test cycles. Changes to the code must be deployed in a controlled manner, in batches, and must be able to be regression tested. Test cycles facilitate this. They show which version of the code generated which particular defects / ran successfully.

A new test cycle cannot begin until the current test cycle is complete. This is to ensure that any defects introduced in the relevant code-drop are discovered in the appropriate cycle, and not confused with a later release.

There are multiple options for the system test of PPL, depending upon timescales involved, for example:

- Have at least two complete test cycles, where all the in-scope tests are run at least once (once in each cycle). A degree of regression testing would be executed at the start and end of each subsequent test cycle to show that the latest code-drop hasn't broken existing functionality.
- If timescales do not permit for multiple test cycles, an alternative (though with greater risk) is to run the test cycle once and to have a comprehensive regression cycle at the end. Contingency time must be allocated in the plan for the likelihood that defects will be found during regression (where bugs have been introduced as a consequence of defect fixes) and their subsequent resolution and retest.

Regression Testing

At the start and end of each new test cycle a regression test must be run. The purpose of a regression test is to show that the latest code-drop has not had an adverse effect on the existing functionality, and where defects are discovered, the relevant affected areas are highlighted. It must be conducted at the start of the cycle to show the functionality from the previous cycle hasn't been broken, and also at the end of the cycle to show that any defect fixes that have been introduced during that test window haven't had an adverse impact.

The regression suite is comprised of a collection of high-priority tests and defect fixes. It is designed to be a series of tests that can be run at will if required, and also without great duration.

System test must have its own test environment, separate to the development and MAT environments. The reasons for this are (but not limited to):

- If development was ongoing on the same code as the system test, the system would constantly be changing, leading to superfluous defects being raised and invalidation of testing
- By the nature of system testing, actively seeking defects may have an impact on the test environment, which would disrupt any other test activities on the shared space
- System testing would not be run in parallel with MAT
- The test data required for some system test scenarios requires manipulation, which would invalidate MAT if it were on the same environment

Entry Criteria

- System test plan (schedule) authored and published
- Dedicated test environment
- Smoke test of test environment prior to commencement of testing
- Development and Ebix system test activities completed
- Ebix to provide system test scripts used during their system test phase
- Ebix to provide defect log of issues found during their testing
- Release notes received with code-drop

- Definitions of user profiles for system access provided
- Test requirements authored
- Test scripts to be authored and to industry-standard format
- Test scripts to be linked to at least one corresponding test requirement
- Defect management process to be agreed and in place
- Test coverage between Ebix and PPL to be agreed
- Priority of tests agreed and approved
- Number of test cycles agreed

Exit Criteria

- All high-priority test scripts to have been executed at least once (mitigation provided where this is not the case – the mitigation will vary depending upon the reason why the test script(s) couldn't be executed)
- Pre-agreed number of test cycles to have been completed
- No outstanding Severity 1 defects from system test
- No outstanding Severity 2 defects from system test
- No more than 10 outstanding Severity 3 defects from system test
- No more than 20 outstanding Severity 4 defects from system test
- System test report authored and published
- Ebix to share its test scripts and test results

6.2.3.2.1. Ebix Responsibility

Ebix will system test the relevant deployed code in their eeTEST environment following completion of development testing. There will be a code freeze prior to Ebix system test commencing to ensure that the relevant scope has been captured and that test plans can be compiled accurately.

Ebix will perform their system test and will only deply to eeUAT for PPL to conduct their testing once they have completed their test allocation.

6.2.3.2.2. PPL Responsibility

PPL will only commence their testing on eeUAT once Ebix have provided the assurance that the release is complete and has been system tested.

Any defects found will be reported to Ebix, where they will be investigated, resolved, run through eeTest and then in to eeUAT.

6.2.3.3 MAT

The focus of MAT is to test the business process that the users of the system would follow. The focus of MAT is very different from previous test phases as it is intended to show that the system works, as opposed to targeting defects.

The test requirements for MAT are taken from the BRDs, in keeping with the business scenario aspect of MAT.

MAT cannot commence until the quality gate from system test has been achieved.

Why Perform MAT?

Essentially, to protect the project (and the company) from damage brought about by releasing a system that has been shown to be functionally correct, but not correct in terms of business use and not integrating with the other systems and processes in use.

MAT will show the impact of a system on the project / company prior to it being deployed to Production.

There may be resource conflicts in seeking testing resource from active business users i.e. asking front-line staff to be made available to perform a temporary test activity instead of undertaking their regular duties. However, this is far less of an issue compared with deploying a system that is damaging, requiring roll-back, re-work, retest and redeployment.

Testers

For PPL, the testers will be comprised of brokers, underwriters and operations in order to accurately reflect the day-to-day business processes expected of the system.

It is essential to note that there must be collaboration between the test groups to ensure that the full E2E process is followed. For example, the brokers must notify the underwriters of the intentions of their test scripts so that the underwriters can author their tests accordingly.

The example below is not a test script, but instead shows the interaction in the process flow between the broker and underwriter. Their tests must reflect this interaction. A requirements / functionality process flow matrix will be communicated to the MAT audience to show which business scenarios and functionality require test coverage. This will also show which actor is involved in each part of the process and will assist in an efficient test execution.

The PPL Test Manager and brokers and carriers will provide support in the set-up and organisation of MAT.

Please note – there is no minimum or maximum numbers of testers required for MAT. The audience is dictated largely by the array of functionality deployed in each release. For example, if a major release is deployed, a large cross-section of the market is required to provide validity, whereas where a small change is deployed (such as defect fixes on a warranty release) the audience will be limited to the testers who raised the original defects.

Entry Criteria

- System Test Quality Gate passed
- MAT Plan authored and published
- Dedicated test environment (browser-based)
- Clear acceptance criteria identified for broker / carrier participants
- Smoke test of test environment prior to commencement of testing
- System test report published
- Release notes received with code-drop
- No outstanding Severity 1 defects from system test
- No outstanding Severity 2 defects from system test
- No more than 10 outstanding Severity 3 defects from system test
- No more than 20 outstanding Severity 4 defects from system test
- Tests to be authored and approved by stakeholders
- Tests to be published to MAT participants
- Defect management process to be agreed and in place
- Priority of tests agreed and approved
- Number of test cycles agreed

- Collaboration between MAT groups organised and planned

Exit Criteria

- All high-priority tests to have been executed at least once (mitigation provided where this is not the case)
- Pre-agreed number of test cycles to have been completed
- No outstanding Severity 1 defects
- No outstanding Severity 2 defects
- No more than five outstanding Severity 3 defects
- No more than 15 outstanding Severity 4 defects
- MAT test report authored and published
- MAT test cycle signed-off by broker / carrier representatives (IT Sub-Committee)

6.2.3.4 Non-Functional Testing

For the catalogue of non-functional requirements in-scope for PPL, please refer to the summary section of Non-Functional Requirements in this document. This states the standards and metrics to be achieved by the non-functional testing.

Once the code has been approved as 'stable' via sign-off from the preceding test phases, tests must be conducted to ascertain that the system can be used by multiple users concurrently at typical (and above) business volumes. Examples of these testing methodologies are:

- Security testing
- Performance testing
    - Load testing
    - Stress testing
    - Soak testing
    - Load balance testing
    - Spike testing
- Fault tolerances
- User interface
- Disaster recovery
- Audit / administration / monitoring
- System requirements

It is important to note that before a non-functional test activity can commence, baseline requirements of system performance must be documented. For example, a stress test that covers concurrent logins must have a requirement that states how many concurrent logins are allowed before system degradation.

Alternatively, performance testing must have a value for the times a new screen takes to be displayed upon a particular action being executed – tests would then be run to show how this value is affected by multiple concurrent users performing the same action.

Non-functional testing will be conducted by both internal Ebix resource and specialist external resource, depending upon:

- The test tools available to the internal resource
- The level and complexity of the testing to be undertaken – this may require specialist skills to code automated tests. Please note that this automated testing is separate to the automated testing that is out of scope for PPL – automated testing in this context refers to the programmatic facility employed to simulate performance variables, and not conduct functional tests.

Third-party resource is also required in order to provide assurance that internal non-functional testing is valid.

Non-functional testing will be conducted in the Production Mirror environment. There are two significant reasons for this:

- Non-functional testing has to be conducted on the most life-like environment possible, in order to replicate the conditions under which it will be used post-go-live. Undertaking testing in a Production Mirror negates any chance of environmental differences
- Non-functional testing will be conducted once system test has completed and, providing that the code is stable enough for non-functional testing, there will be no other users on the Production Mirror environment, negating any issues with environment downtime inevitably caused as a consequence of non-functional testing

This section will refer to the methods of non-functional testing that will be utilised.

6.2.3.4.1. Security Testing

Access to the system must be tested to ensure that only authorised users can login. Following-on from that, assurance must also be provided to show that once the authorised user has entered the system, that they can only view / access the functionality / data for which they have permission.

For example, to access PPL, each user will have a unique login username and password and two factor authentication, via a PIN code and memorable word. Tests will be authored that cover login via incorrect password, mixed-case username (where the username has to be case-dependent), correct username and login but incorrect PIN, and so on…

At the time of writing (prior to development completion) it has still to be confirmed whether PPL will support login access via a client's home application (such as an AON user logging-in to its parent application and then accessing PPL via a link without the need to conform to standard PPL security) or solely through the PPL login procedures, or the ability to support both options.

Other aspects (but not limited to) of Security Testing that must be in-scope of non-functional testing:

- SQL Injections
- Penetration Testing

SQL Injection

This is a code injection technique designed to see if, for example, malicious SQL submitted into the login field will have a negative impact on the system – whether it be to crash it or exploit a security flaw to dump the database.

Penetration Testing

This is a simulated attack on the system that looks for security weaknesses in an attempt to gain access to the system. The penetration testing will ascertain whether the system is vulnerable to attack, if the relevant defences are sufficient, and which defences hold or are defeated.

6.2.3.4.2. Performance Testing

The performance of the system must be tested to provide the assurance that it responds in accordance with predefined criteria such as expected performance times for system start up, screen to screen refresh, saving data, document upload, and so on. This also covers the scalability aspect of the system – for example, the platform is rated to support 9,000 registered users, of which 3,000 will be regular active users. Ebix does not anticipate any constraints supporting a userbase of 10,000.

Stress Testing

This shows the upper limits of the system capabilities under times of extreme usage. For example, where the system is rated to handle 10,000 active users, stress testing will simulate all 10,000 logging-in over a similar time period and / or performing critical business transactions concurrently.

Load Testing

This will show any bottlenecks in the system and to understand the behaviour of the platform under a specified load. Response times will be monitored for the relevant transactions / processes under the load to show any weaknesses.

Soak Testing

This is the process of running the platform under constant heavy load to show that the system can handle continuous high-volume usage at its prescribed limits of user activity. For example, 5,000 registered users concurrently running transactions through the system for a set period of hours – employing a significant load for a significant amount of time.

Load-Balance Testing

Where a system has a load-balance configuration, this must be tested to show the effects on the platform where the load-balance is changed – for example, if it is a simple 50/50 balance between two servers, what happens when one of the servers has a reduced capability, or is unavailable?

Spike Testing

This is where an unusually high volume of users all perform the same process simultaneously. A sudden increase in the load on a piece of functionality will show whether the system can handle it, where performance suffers, or whether the sudden increase in usage has no impact.

6.2.3.4.3. Fault Tolerances

Fault tolerances are tested by simulating a system or application failure and monitoring the expected reporting of the issue and the recovery.

Fault monitoring must be accurate and the recovery of the system post-failure must be within the tolerances agreed in the non-functional test requirements.

For example:

- What reports are sent to the administrators upon system failure?
- How soon are the failure reports sent to the administrators?
- How are the users notified of the system failure?
- How soon from the system failure can the system be accessed again?
- Upon system recovery, do the users carry on from where they left off, or do they have to start the relevant process they were involved in again?

6.2.3.4.4. User Interface

Non-functional testing of the UI is to show that the front-end performs as specified other than for functional processes. For example, if a client opens multiple sessions on a single browser session (such as multiple tabs) and attempts to perform simultaneous actions on the same transaction, the test will aim to show that the user cannot undermine the correct process of the system.

This will also cover (but not limited to):

- Screen resolution size (are any buttons / links truncated or lost depending on reduced screen size)
- Accessibility standards (such as in accordance with the DDA if appropriate)
- Branding
- Timezone display
- Monetary value display (comma separation, currency etc)

6.2.3.4.5. Disaster Recovery

Disaster Recovery (DR) testing is separate to fault tolerance testing. DR is defined as the processes required for recovery or continuation of the critical business system / function following the loss of the primary server / building in the event of natural / human-induced catastrophe.

Therefore, the DR tests must simulate such a catastrophe and whether the recovery is within the tolerances specified in the non-functional requirements. This will require testing not just from an application viewpoint, but also in determining the effect on the clients.

For example, a DR test will examine whether the secondary server picks-up from the loss of the primary server within the specified timescales and at which point in the transaction process is the data recovered.

Two aspects will be defined and tested:

- Recovery Point Objective (RPO)
- Recovery Time Objective (RTO)

The RPO is the point at which, when the platform is recovered, the data is recovered to. For example, if the RPO is defined as 30 minutes, when the platform is returned to usage the data in the system will be no more than 30 minutes aged from the point when the platform was lost.

The RTO is the timeframe in which the platform must be restored. For example, if the RTO is defined as four hours, the platform must be restored for usage no more than four hours from the moment it was lost.

6.2.3.4.6. Audit / Administration / Monitoring

Audit

The audit trail must be tested to show that it is an accurate reflection of the transactions processed through the application. There are two elements to the audit testing of PPL:

- There is an on-screen option for the users to view the audit trail for the relevant transaction via the BEV
- Administrator view of audit trail

Additionally, there are other non-functional requirements that need testing for scenarios such as where the transaction is cancelled (the audit trail will not be retained).

Administration

The administration aspect of the system requires thorough testing, both in terms of the administration role that Ebix provides and what (if any) functionality the customer administration will have.

For example, the tests will be required to cover the following functionality at a minimum:

- What permissions the admin can grant to an individual user role
- Ability to over-ride an action performed in error by a client
- What user roles are available to the admin to assign to clients
- The process for disabling a user's access – both client app access and through individual browser access

Monitoring

Management Information (MI) is a significant aspect of the platform. Multiple non-functional requirements cover this piece. Tests must cover, at a minimum:

- System health monitoring – accuracy
- System health monitoring – alerts and notifications (which will require simulated events to generate the alerts)
- Scheduled batch processing for the MI reports

- Back-up details
- Any conflict between MI reporting and business transactions

6.2.3.4.7. System Requirements

The PPL application has a minimum set of operating requirements.

Tests must be authored to cover the non-functional requirements for this. As an example:

- Which browsers are supported
- Which version of supported browsers are supported
- Is tabbed browsing permitted
- Is access via a mobile phone or tablet permitted
- What screen resolutions are supported

6.2.3.5 FAT

This is the test activity conducted immediately prior to go-live. Its purpose is to show that the critical functionality operates as expected in the Production environment. It is not intended to be a comprehensive test of the functionality, but to show that the packaged build has been deployed successfully to live.

In essence, it is almost 'go-live', as the code is deployed to the Production Support environment for the purpose of the test. Once the Pre-Prod testing has been shown to be successful, the code will be deemed fit for go-live, where a further smoke test of critical functionality will be executed as a final confirmation in Production.

The Pre-Prod tests are essential for the purposes of regression testing for the clients who are currently using the existing live aspects of the overall PPL infrastructure and, critically, the configuration of users set up on the platform. Pre-Prod will be as close to live as possible and will therefore require a cut of live data in it in order to facilitate the regression testing required by the client.

Post-pre-Prod testing, typically, if the formal go-live date is a Monday, the code will be deployed to Production over the weekend before and tested prior to the client viewing it.

The tests will be designed to cover critical functionality and any high-severity defect fixes, and will be executed by either the test resource or SME.

If the Field Acceptance Test fails, a pre-determined roll-back procedure must be followed to return the Production environment to its previous version so that the end-user sees no difference.

Entry Criteria – Pre-Prod tests

- MAT phase signed-off
- Non-functional test phase completed and signed-off
- Go-live date agreed
- Code packaged and deployed to Pre-Prod
- Cut of live data deployed to the test environment
- Single test cycle agreed and approved
- Dev / IT support in place during FAT window
- Contingency days agreed in case of critical defects found during regression testing
- Communication plan for users (and for what happens if it goes wrong) for go-live
- Critical tests identified
- Roll-back procedures in place

Exit Criteria – Go-Live tests

- All agreed tests executed successfully
- FAT report authored and published

6.2.4 Test Requirements

For every test phase post-unit testing, a test must have a corresponding test requirement. The test requirement is a test condition derived from the functional documentation that is granular enough to be tested in isolation. One statement in a functional document may generate several test requirements. For example, a statement of "the telephone must be capable of accepting 12 characters" could create the following test requirements:

- Confirm that the Telephone field can be submitted containing 12 numeric characters
- Confirm that the Telephone field cannot contain more than 12 numeric characters
- Confirm that the Telephone field cannot be submitted if it contains any alpha characters
- Confirm that the Telephone field cannot be submitted if it is empty
- Confirm that the Telephone field cannot be submitted if it contains special characters
- Confirm that the Telephone field can be submitted with at least one number

It is essential that the test requirement references the source document. If a defect is raised against the corresponding test script, the source of the test will require to be identified to confirm whether or not the test is valid.

This process forms the Requirements Traceability Matrix, which is used to show the level of test coverage for a system against the source documentation. For example, it can be submitted to business stakeholders to show::

- The level of test coverage against the source documentation
- Test requirements

Business stakeholders can review the test requirements to ascertain if any functionality has been missed, or to provide sign-off if they approve the test coverage

6.2.4.1 Priority

Test requirements are grouped into priority order. The reason for this is that it is not typically enough time for it to be possible to test every variation that a system can facilitate, and a decision must be made as to which requirements are to be tested first, and which ones have to be granted a lower priority with the knowledge that they might not be tested, depending upon timescales.

6.2.5 Stakeholder Sign-off

Prior to any test phase commencing (other than unit / integration test), stakeholder sign-off of the test coverage must be granted. The reasons for this are:

- Stakeholders have the assurance that they know what is being tested
- It is not left to the test team to decide the approval of the test coverage
- Where any test coverage concerns are raised during the latter stages of testing or upon go-live, the test team are absolved. It is not the test team's responsibility to sign-off test coverage

6.2.6 Test Scripts

Test scripts (or test cases) for PPL must confirm to industry-standards. It is assumed that Ebix and PPL will be using different test governance tools. The tests must still contain the same structure:

- Test name
- Link to a test requirement

- Pre-requisites listed
- Test data requirements listed
- Success criteria listed
- Purpose of test
- Test step ID / Action / Expected Result

6.2.7 Test Execution

Regardless of the test tools used by Ebix and PPL, the tests must be executed in accordance with acknowledged industry standards.

All tests must be executed against a corresponding test script. If a new test is formulated during test execution (as inevitably happens) a test script can be authored and retrospectively linked to a test requirement.

During execution, all tests must conform to the following metrics:

- Capture the version of code the test is being run against
- Executer's details
- Time and date of execution
- Execution duration (where possible, depending upon the test tool in use)
- Linked test requirement
- Success / failure of each test step
- Comments added against relevant test steps where applicable
- Overall success / failure of the test script
- If a test script can't be completed, an explanation must be provided
- Evidence (screenshot or data file, for example) captured for each success criteria
- Defect evidence and steps-to-reproduce where a bug is encountered

Collaboration

System testing will be a collaborative effort between PPL and Ebix. Test requirements authored by both teams will be shared at the earliest opportunity to ensure the necessary level of test coverage is provided and that there is no duplication of effort. Upon agreeing the segregation of the test effort between the two teams, each team will then author their allocation of test scripts in isolation and regularly provide updates on progress.

The execution of these tests will be performed by the respective teams that authored the test scripts. This will negate misunderstandings / misinterpretations. Daily reporting of test execution progress will be conducted to provide the project with an overall picture of test progress.

For MAT, collaboration will be required between the test resource (brokers / underwriters / carriers) to ensure that each parties tests link together to provide a full E2E test coverage. The PPL Test Manager will assist in organising the test execution so that it is as seamless as possible. A decision will be taken as to whether it can be organised to get all test resource for MAT together for the testing (as the platform is web-based, location may be irrelevant). This will greatly increase collaboration and efficiency.

7 TEST PLAN

The natural progression from the test strategy is the test plan. The test plan, in this context, is the schedule of testing. This will be published separately to the test strategy.

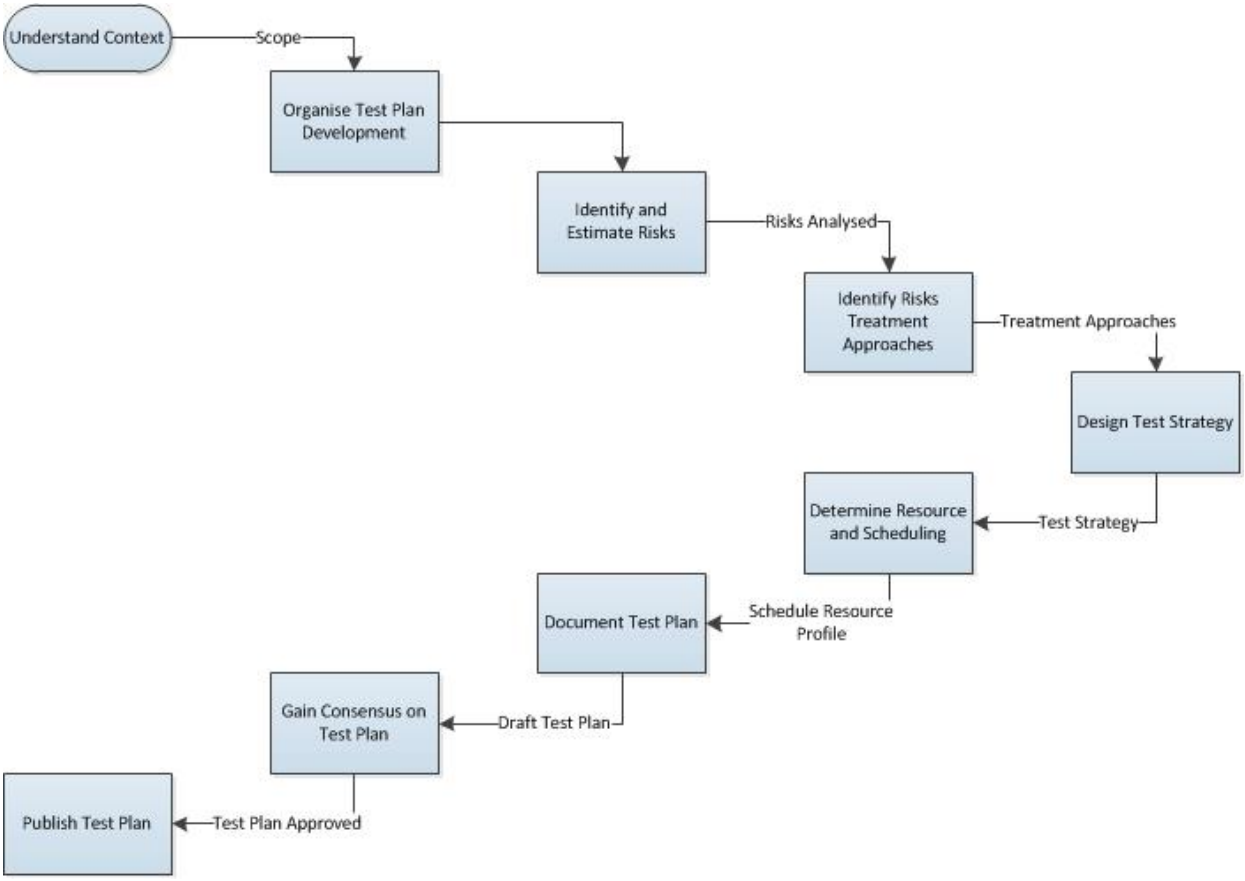

Figure 5 - Test Plan design progression

The test plan will document the following:

- Resources involved in the testing (at each stage)
- Timescales of the test window
- Code-drops
- Test phases
- Test cycles
- Regression test cycles

The test plan will be shared with all parties to show who is required when, and what functionality will be tested at any given date.

It is important to note that the test plan will not be written at a granular level. For example, it won't aim to state exactly which test script / how many will be executed on any given day. Testing is too unpredictable for a test plan to ever accurately reflect this reality. Instead, functional areas will be shown explaining what is being tested over each relevant time period. Estimates of the duration of the execution of each functional area will be ascertained by the Test Managers of PPL and Ebix and factored into the test plan.

The example snap-shot test plan for system test below (purely for illustration purposes) is intended to show how the test effort will be structured – who is testing what functionality on any given day and for how long. Some areas of functionality will take longer to execute than others, so it may well be the case that only one area gets tested one day at a time if it is labour-intensive.

	Date						
Functional Area	**07 December 2015**	**08 December 2015**	**09 December 2015**	**10 December 2015**	**11 December 2015**	**12 December 2015**	**13 December 2015**
Open-market placements							
Insurance							
Reinsurance							
Multi-section risks							
Placing Binder/Lineslip/Facility							
Placing Declarations under Binder/Lineslip/Facility							
Initial submissions							
Quotations							
Firm-order placements							
Completing risk placement							
Code version:	PPL v1.001	PPL v1.001	PPL v1.001	PPL v1.001	PPL v1.002		
Planned code-drop							
Key:	LMG						
	Ebix						

Figure 6 - Example System Test Plan

A master test plan per class of business will be published, which will incorporate the dates for the individual test phases.

8 TEST DATA

8.1 Availability of Test Data

For system test, the test environment will not be pre-populated with life-like test data. Ebix has confirmed that it will pre-populate the test environment with a limited amount of test data. The testers will generate their own test data as required for bespoke test scenarios through the natural progression of test execution.

The test data that will be on the system test environment will be provided by Ebix as follows, for example:

- Test broker accounts
- Test carrier accounts
- User profiles for brokers and carriers
- Risk codes
- Templates

For MAT, the test data supplied will be the same base details as that provided for system test. The difference for MAT will be that the test accounts / profiles will be based upon the MAT testers' live profiles. Any risk of sensitive client data being available to users who aren't authorised to access it is mitigated by the user profile / permissions testing that will have been tested and confirmed in the earlier system test phase.

For system testing and MAT, the testers will rely solely on user-generated test data, which will be more suited to the bespoke test scenarios naturally required during system testing.

The client users will also generate their own test data suited to their tests. Details of the test data submitted must be recorded as, for example, a broker submitting a risk to the system will need to notify the underwriter of the relevant details so that they can confirm it appears correctly.

8.2 Data Sanitisation

For MAT, there is no requirement for data sanitsation. System test will provide the assurance that user profiles / permissions works as specified, and only those users with the correct permissions to view the relevant data will be able to do so.

9 DEFECT MANAGEMENT

9.1 Overview of Defect Management

Defect management in a test activity is the process by which any potential defects are:

- Raised by the originator
- Assessed by an assigned triage group (PPL Test Manager plus Ebix counterpart, with Ebix and PPL BAs consulted where needs be)
- Assigned to the relevant resource for resolution
- Re-assigned to the originator for verification
- Reported upon

Due to the collaborative nature of the test effort between PPL and Ebix (Ebix will have the defects assigned to them for resolution, as code development is their responsibility) it is unlikely that an integrated defect management test tool will be able to be used.

9.2 PPL Defect Management Process

In lieu of an integrated test tool, any defects found by PPL will be documented on the project's defect-tracking tool (Google sheets) to keep a record of it through to resolution. It is essential that progress is reported to PPL in the meantime. Defects found by Ebix will be recorded directly into their defect management tool.

Upon resolution, Ebix will inform PPL that the bug has been fixed and is ready for retest. PPL will then amend the status of the defect ticket in its tool and test the fix and update the ticket accordingly. A daily defect report would be generated to show the status of defect progress to all relevant parties.

The same process will be used for system test and MAT, but with the PPL Test Manager acting as a liaison point between the MAT resource and Ebix. Please see the Communication Plan section of this document for details of this process.

In any defect management process discipline is essential, and all defects must be raised and recorded in the same way, with the same process. Any defects raised outside of the agreed process are deemed not to exist, and will not (and cannot) be dealt with.

9.2.1 Actors

9.2.1.1 System Test

- Testers – this will be a combination of test resource derived from the PPL and Ebix test teams
- Triage – the triage point for new defects will be the respective test manager of the test team that raises the defect
- Development Manager – this is the Ebix development manager who will assign the defects to the relevant development resource
- Development – this is Ebix development resource who are responsible for defect resolution

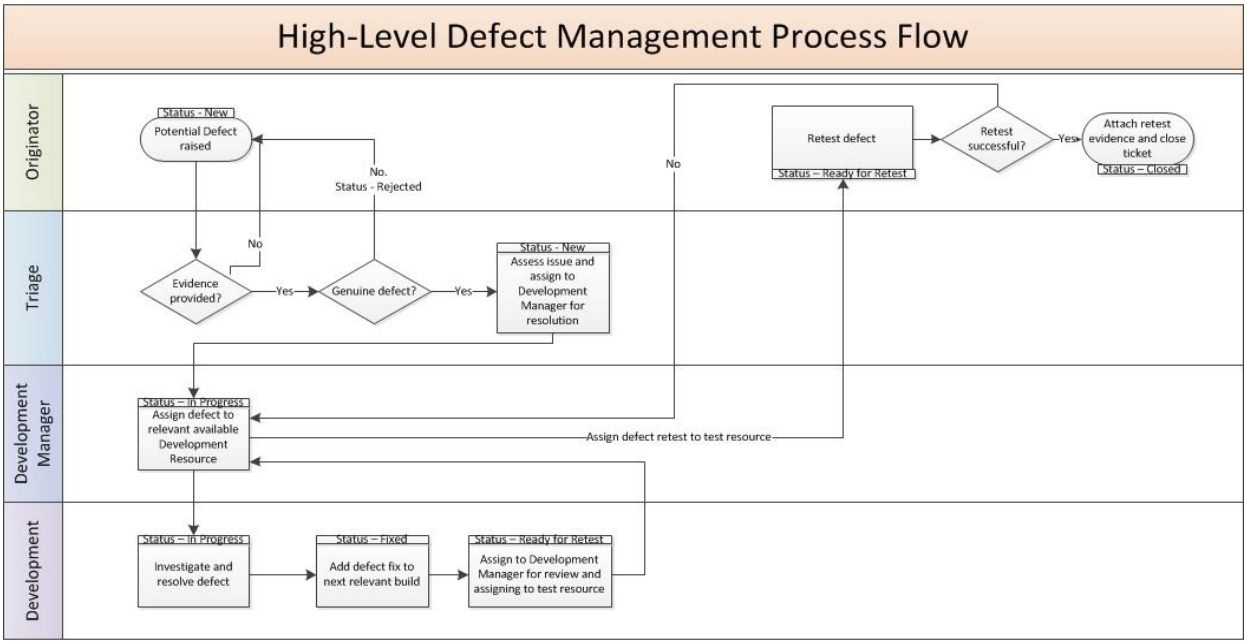

Figure 7 - Defect management process

The example above shows the logical defect management process. Depending upon the chosen defect tracking tool, the status of the defects may differ, but the process will be the same:

- Tester raises a defect
- Triage point asses validity and either rejects it or assigns it to the development manager
- The development manager assesses which development resource is best-placed to investigate and resolve the defect
- The defect is fixed and assigned back to the development manager to ensure the correct status is recorded, evidence of fix is attached, and then it is assigned back to the tester who raised the defect
- The tester then tests the defect fix and reports the success / failure (with the process beginning again if the fix has failed).

9.3 Standards

Regardless of the defect management software in use, a defect ticket must have the basic industry-standard metrics as a minimum:

- Summary of defect (a concise defect title)
- Detailed description of the defect – including what was expected and what happened, including steps to reproduce
- Screenshot / evidence of the defect

- ID of the defect originator
- Version of code the defect was raised against
- Severity of the defect
- Priority of the defect
- Environment in which the defect originated
- Time and date the defect was found
- Application area (where applicable) – eg Quote or Firm Order
- Test Cycle in which the defect was found (where applicable)
- Test Phase (if multiple test phases are on the same environment)

It is inevitable that during the MAT phase, the same defect will be raised to the PPL Test Manager by multiple organisations. This is where the triage role of the PPL Test Manager is paramount. The defect-tracking tool has an in-built search facility which will be used to show if a similar / same defect has been raised before. The outcome of this, and the progress of the relevant defect, will be reported back to the defect originator.

9.4 Service Level Agreements (SLAs)

In terms of defect management, SLAs are not yet agreed between PPL and Ebix.

However, it would be logical to assume that high-priority / severity defects are resolved as a critical priority at the expense of lower-priority bugs. For critical defects that prevent testing, it is also assumed that, regardless of any SLA, these would be resolved as urgently as possible.

9.5 Defect Severity

The following defect descriptions / severities are industry-standard, and would be assumed to be used by both PPL and Ebix, or a variation of it (depending upon the defect management tools in use):

9.5.1 Severity

This is the extent which the defect can affect the software. It defines the impact that a given defect has on the system. For example, if by clicking on the Login button of a system causes it crash, this is a high-severity defect.

Severity Rating	Description
1	**Critical**: The defect results in the termination of the complete system or one or more components of the system and causes extensive corruption of the data. The failed function is unusable and there is no acceptable alternative method to achieve the required results. Therefore the severity will be stated as critical. Can't go-live with this unresolved. Must be fixed immediately.
2	**Major**: The defect affects major functionality or critical data. It does have a workaround, but it is not obvious and is difficult. Can't go-live with this unresolved. Must be fixed as a priority unless any Sev 1 defects exists.
3	**Moderate**: The defect does not result in the termination, but causes the system to produce incorrect, incomplete or inconsistent results, but has a straight-forward workaround. Therefore the severity will be stated as moderate. Can't go-live with any Severity 3 defects.

4	**Minor**: The defect does not result in the termination and does not damage the usability of the system and the desired results can be easily obtained by working around the defects. Therefore the severity is stated as minor. Can go-live with a negotiated amount of Sev 4 defects – but this must be agreed with the business.
5	**Trivial**: The defect does not affect functionality or data. It does not require a workaround, and does not impact productivity or efficiency. Trivial defects will only be fixed if all other defects have been remedied or can be done as part of a fix to a higher-severity defect.

10 ASSUMPTIONS

The following assumptions have been made about the test activity for

ID	Summary	Description	Owner
01	Sufficient documentation will be made available by Ebix to facilitate test authoring.	Ebix will provide sufficiently detailed documentation with which to author tests from for the proposed functionality and the non-functional requirements. A user guide(s) will be provided to show existing functionality, which facilitate the authoring of regression tests.	Ebix
02	Development Testing completed before system test starts	Before system testing can commence, development testing must have completed successfully and PPL informed accordingly.	Ebix
03	System test environment availability	The PPL and Ebix test teams require access to the system test environment as soon as possible in order to familiarise themselves with the new functionality and navigation, and also to conduct the system test once the solution is deployed. It is assumed that the PPL team will provide its own smoke test prior to full system test commencement.	Ebix
04	Dev support	Ebix development support will be required for the duration of the test effort in order to provide defect resolution and technical answers to any queries raised.	Ebix
05	No test data pre-populated on system test environment	It is assumed that the system test environment will not be pre-populated with business-like test data and that testers will have to generate their own bespoke data. However, whilst there will be no business-like test data, there will be dummy broker / carrier profiles set up, along with risk codes and templates.	Ebix

06	MAT environment not pre-populated with a cut of live	The MAT activity will not require simulated data (other than user profiles) to be available in advance.	Ebix
07	Test requirements will be shared between Ebix and PPL	Collaboration will take place between Ebix and PPL to author test requirements. Areas of functionality will be divided amongst the two teams and test requirements will subsequently be authored accordingly.	PPL and Ebix
08	Joint effort between Ebix and PPL for system test script authoring and execution	Following-on from the collaborated test requirement authoring, Ebix and PPL will author test scripts against their respective test requirements and provide regular updates on the completion path.	PPL and Ebix
09	MAT environment availability	The PPL team require access to the test environment as soon as possible in order to familiarise themselves with the new functionality and navigation, and also to assist in the construction of training material. It is assumed that the PPL team will provide its own smoke test prior to full MAT commencement.	Ebix
10	Brokers and carriers will provide MAT resource	LIIBA, LMA and IUA will advise who will participate in the MAT.	LIIBA, LMA and IUA
11	MAT will be integrated between brokers and carriers	The MAT activity will be an integrated effort between all parties involved. This will provide a genuine E2E aspect to the test. PPL will work with LIIBA, LMA and IUA to assist in co-ordinating this. The functional areas in-scope of the MAT will be arranged into high-level test scripts by PPL to assist in the testing scope.	PPL, LIIBA, LMA and IUA
12	Quality gates will be adhered to	Before any test phase can commence (post-unit and integration tests) the quality gates for each phase must be met.	PPL and Ebix
13	Release notes will be provided	Ebix will provide release notes with every code deployment to show the functionality included / removed / amended and defect fixes included. For the MAT phase, PPL will relay the release notes to the test resource to ensure that they are aware of functional changes.	Ebix (and PPL)
14	Test reports will be generated upon completion of each test phase	The test manager responsible for each test phase will produce a test report to show that the test phase has completed, including relevant test metrics such as number of tests executed, defects raised / fixed etc.	PPL and Ebix

15	System test will incorporate a full E2E capability	System testing will commence on the complete, integrated, platform with all specified functionality available to test.	Ebix
16	MAT cannot start until system test completion	In accordance with the quality gate methodology, MAT will not commence until system testing has successfully completed.	PPL
17	FAT cannot start until MAT completion.	The Field Acceptance Test cannot begin until the assurance that the system is fit for purpose has been achieved. This means that FAT will only commence once MAT is signed-off.	PPL, LIIBA, LMA and IUA
18	Non-functional testing will be conducted on the Production Mirror environment.	Non-functional testing must be conducted on the most life-like environment possible in order to accurately replicate the conditions the application will be used in once it has gone live. Non-functional testing will occur once system testing has completed and the code can be assumed to be stable (eg. no major changes to the functionality). No other users will be testing in parallel on the Production Mirror environment.	PPL

11 RISKS AND MITIGATION

The following risks have been identified for PPL and must be acknowledged. Mitigation has been provided where applicable.

Risk ID	Description	Impact	Severity	Mitigation	Owner
01	Functional documentation not made available.	Lack of functional documentation severely impacts the ability to author test requirements and test scripts. Potential loss of confidence in the system if built in the absence of functional documentation.	High	BRDs have been made available for the CRs generated out of the Practitioners List out of the Terrorism release – something similar may occur for future releases. However, Ebix have had enough advance notice now that there is no reason that functional specifications cannot be provided as development commences. A user guide will be issued that can be used to author the regression tests for the existing functionalty. Ebix has agreed to make a test environment available which will allow testers to fill in any blanks in their understanding of navigation / processes.	Ebix
02	System test environment not available in time to provide a sufficient test prior to MAT.	Unavailability of the system test environment on the given start date of system test will delay testing and impact the test plan and potentially result in not all of the planned tests being executed and / or delay to go-live. Ideally, the system test environment is required some days in advance of the planned	High	Ebix has agreed to provide PPL with access to the system test environment upon completion of development, in advance of the start of testing.	Ebix

Risk ID	Description	Impact	Severity	Mitigation	Owner
		start of testing to allow for smoke testing.			
03	More defects discovered in system test than expected / critical defects hindering testing.	Finding significantly more defects than expected, or several critical defects that impact test execution, will have an impact on the test timelines and potentially reduce the number of planned tests that can be executed and / or delay to go-live.	High	Development testing will be completed prior to system testing commencement. These tests will provide the assurance that the code executes and negates the chances of critical defects that prevent testing being found. A planned smoke test of the system test environment prior to its commencement will also show that critical functionality executes. If critical defects are found during the smoke test, a window exists in which to correct the issues prior to the start of system test.	Ebix and PPL
04	Lack of co-ordination in MAT.	If the MAT is not co-ordinated correctly, the respective tests authored by the brokers and underwriters will not integrate and the test effort will be inefficient.	Medium	PPL will co-ordinate the test effort to ensure that the MAT tests dovetail according to the relevant business processes. A traceability matrix will be produced to show what functionality is required to be tested and where it links to the next stage in the process.	PPL PM and brokers and carriers
05	Lack of broker / carrier engagement	A lack of participation across the broker / carrier community creates a risk of diminishing the validity of the MAT.	Medium	Carriers and brokers will be engaged as soon as possible by PPL to ensure optimum take-up in MAT. Benefits such as an early look at the platform and the chance to get familiar with it prior to go-live will be explained.	PPL, LIIBA, LMA and IUA

Risk ID	Description	Impact	Severity	Mitigation	Owner
06	MAT will be limited to testing new functionality and performing a small regression test on existing functionality.	Due to the limited timescales for MAT, the test effort will focus on the new business processes and functionality, as opposed to a system-wide test, and will also include an element of regression testing of existing functionality.	Medium	A MAT activity is intended to be a user test of the business processes, and not a functional test of the system. The assurance that the platform is functionally sound will have been provided by the system test, and therefore negates the need to do a comprehensive functional test in MAT. Users new to the system will be required to perform a wider test in order to mitigate their initial lack of knowledge of the system.	PPL and brokers and carriers
07	Scope creep.	As testing continues, there is a risk that the original scope of the release is expanded. This will require more testing than planned (truncating test effort or extending deadlines) and will require more regression testing.	High	Any additional testing to that in the original plan will require scoping and replanning. If timescales are to be maintained, lower-priority tests can be de-scoped in order to maintain sufficient coverage. If lower-priority tests cannot be sacrificed, timescales will be amended accordingly to fit the extra test effort.	PPL and Ebix
08	Non-functional testing identifies architectural issues.	Any critical issues discovered in non-functional testing could potentially require architectural changes. As non-functional testing occurs during the latter stages of the test effort, this does not provide much time to resolve them, retest the issue, and also regression test to ensure that the functionality of the platform has not been adversely impacted.	High	The non-functional requirements have been made available to Ebix prior to development completion. Once the responses are received by PPL confirming what can be achieved, these responses will be tested.	Ebix

Risk ID	Description	Impact	Severity	Mitigation	Owner
09	MAT resource test outside of the defined scope.	There is a risk that MAT resource will engage in their own un-scripted tests. This can lead to spurious defects being raised, as the issues are likely to be invalid business processes / lack of understanding of the new functionality.	Medium	Acceptance criteria will authored by PPL to show the high-level functionality and business processes to be tested. Existing functionality will be regression tested, but not as a functional test – just the business process. Assurance on the quality of the functionality will have already been provided by the system test phase.	PPL and brokers and carriers
10	Late-running development / test phases require that test phases are truncated, overlap, or component parts of the platform are tested in isolation when they become available.	Where test phases overlap, there is a significant risk that a defect found in one of the phases will invalidate the testing in the other. This is why it is good practice to avoid overlapping phases. A truncated test phase means that not all planned tests can be run, or at least not run as many times as planned. Testing functionality in isolation does not provide the correct integration coverage. Whilst each component part may operate correctly on its own, it may fail when integrating to the wider platform.	Medium	Tests are prioritised to show which ones require execution above all others. This ensures that in the worst case scenario, the minimum requirement of tests are executed. Where timescales dictate that some functionality is tested in isolation, the integration aspect will be tested as part of the regression test effort towards the latter stage of system test. If timescales dictate that test phases overlap (system test and MAT), the only functionality that will be made available to the MAT resource will be that which has already been system tested and passed previously. No untested functionality will be made available to MAT.	PPL
11	Non-functional testing causes environment downtime.	Non-functional testing is expected to cause environment downtime, impacting users	Low	Non-functional testing will be conducted on the Production Mirror environment at a time prior to go-live where no other users will be	PPL

Risk ID	Description	Impact	Severity	Mitigation	Owner
		testing in parallel on the same environment.		testing in parallel on the same environment. No environment downtime will occur.	

12 COMMUNICATION PLAN

12.1 Test Reports

Every PPL test phase will complete with a published test report. This will document what testing took place and to provide the recipients with the assurance that the application has met its quality gate exit criteria.

At a minimum each test report will document the following:

- Number of planned tests to be executed
- Number of tests executed
- Mitigation for any tests that weren't executed
- Number of test cycles planned
- Number of test cycles executed
- Mitigation for any cycles not completed
- Metrics of defects found and resolved
- Details of any defects carried-over into the next test phase
- Confirmation of entry criteria met before commencement
- Confirmation of exit criteria met upon completion

For development test phases, Ebix will not produce a test report, but instead provide email confirmation that the relevant functionality specified for the system has been developed and tested.

The distribution list for each test report will differ for each test phase. For example, prior to the start of MAT, the System Test Report will be made available to the brokers / carriers.

12.2 Defect Meetings

The audience and frequency of defect meetings will remain during system test and MAT phases.

12.2.1 System Test

Once system test is underway, the defect meetings will be held daily.

The audience for the meetings will be:

- PPL Test Manager
- Ebix Test Manager
- PPL BA
- Ebix BA
- PPL Project Manager
- It is important to note that the meeting will concentrate as much as possible on the progress of the defect resolution, as opposed to describing the details of each defect – this will have already been taken care of with the correct level of detail entered in the defect ticket and review by the triage point.

12.2.2 MAT

As with system test, the frequency of the defect meetings for MAT will be daily. The defect log will also be published to the associations upon request.

The audience for the MAT defect meeting will be:

- PPL Test Manager
- Ebix Test Manager
- PPL BA
- Ebix BA
- PPL Project Manager

12.3 Regular Progress Reports

For both system test and MAT, the PPL Test Manager will provide a daily report to the stakeholders of the test progress. This report will include the following metrics:

- Test phase
- Test phase cycle
- Number of tests executed
- Progress against expected number of test execution
- Number of tests left to run in the relevant test cycle
- Number of unresolved defects
- Severity and Priority of unresolved defects, and their impact on relevant parties
- Key highlights / achievements
- Details of any impediments to test

12.4 Sign-off

The individual test report per test phase provides the sign-off for that phase to confirm that the testing has completed and met its quality gate exit criteria.

The reports are reviewed by the PPL project manager, who then has the authority to commence the next test phase.

13 DISTRIBUTION LIST

The audience of the test strategy is categorised into those who are required to formally approve and sign-off the document and those who are included in the distribution for information only.

Separate test phase-specific test strategies will be authored and distributed for the relevant participants.

13.1 Required for Approval

- Colin O'Malley – Programme Manager
- Ben Howell – Project Manager

13.2 For Information Only

- Ebix – software vendor
- Kim Darrington – IUA
- Louise Day – IUA
- Adrian Thornycroft – Programme Director of TOM

- Peter Holdstock – LMA
- Jackie Hobbs – LIIBA