<Company> Systems, Inc.

Intelligent Automation

Bot Quality Assurance & Testing Cycle

Confidential

Versioning

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 26 Sept 2018 | 1.0 | Discussion version – Robert, David, Srinivas | Mayank Patel & Shubhen Sarangi, Datamatics |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Introduction 4](#_Toc528370802)

[End to End Testing Implementation 4](#_Toc528370803)

[QA Planning Best Practices 4](#_Toc528370804)

[Test Strategy 4](#_Toc528370805)

[Test Plan 5](#_Toc528370806)

[Test Case 5](#_Toc528370807)

[Defect Management 5](#_Toc528370808)

[Traceability 5](#_Toc528370809)

[Intelligent Automation Testing Approach 6](#_Toc528370810)

[Testing Environment 7](#_Toc528370811)

[Dev Environment 7](#_Toc528370812)

[QA Environment 7](#_Toc528370813)

[Staging and UAT Environment 7](#_Toc528370814)

[Production Environment 7](#_Toc528370815)

[Types of Testing 8](#_Toc528370816)

[Functional Testing 8](#_Toc528370817)

[Integration Testing 8](#_Toc528370818)

[Regression Testing 9](#_Toc528370819)

[Non Functional Testing 9](#_Toc528370820)

[Performance Testing, Load Testing & Stress Testing 9](#_Toc528370821)

[Security & Vulnerability Testing 10](#_Toc528370822)

[High Availability (HA) and Disaster Recovery (DR) Testing 10](#_Toc528370823)

[UAT (User Acceptance Testing) 11](#_Toc528370824)

[Test Data 11](#_Toc528370825)

[QA Sign-Off Report 11](#_Toc528370826)

[Testing Tools 11](#_Toc528370827)

[QA & Testing for other Intelligent Automation Technologies 11](#_Toc528370828)

[Appendix A: Defect Log template 12](#_Toc528370829)

[Appendix B: QA Test Strategy 13](#_Toc528370830)

[Appendix C: QA Test Plan 14](#_Toc528370831)

# Introduction

Intelligent Automation requires sufficient attention to software quality practices to ensure planned business results and technically suitable Bot scripts and deployments. Quality in the Intelligent Automation context applies to:

1. Planned ‘To Be’ business process functionality
2. Targeted business systems and applications that Bot interacts with via UI (user interface) or other surface automation techniques
3. Bot scripts and Bot execution infrastructure/platform

As with other Enterprise Software, Intelligent Automation requires a standardized set of Testing and QA processes, Choice of Relevant Tests & Testing Approaches and suitable testing environment to ensure thorough and rigorous testing. This will ensure that Bots and automated processes are free of errors of both functional and non-functional nature thereby contributing to robust Bots that will ensure minimal disruption to automated processes before deployed in production.

# End to End Testing Implementation

Implementing an end to end testing of robotic process automation (RPA) implementations will help <Company> benefit from reduced turnaround time, reduced re-work, higher efficiency and better accuracy in its planned automations.

QA testing approach should be integrated into the overall process selection, solution design and Bot development approach as indicated below:

1. Identify repetitive tasks performed by the business owner that can be optimized by implementing Intelligent Automation as part of test cases
2. Identify the manual or automated testing requirements for each process.
3. Select those applications, actions, output that are impacted or generated by the intelligent automation rather than testing entire process
4. Enable end to end automation scripting by combining various automation tools for complete testing coverage
5. Produce a single automation script from creating/inputting test data to Bot execution and validation across impacted applications/systems
6. Reuse the automation script and test in higher environments like Staging (Pre-Prod) or UAT (User Acceptance Testing) environment.

# QA Planning Best Practices

It is recommended that QA approach be a deliberated and planned using standard software QA planning practices. Key elements of such practices include:

## Test Strategy

The test strategy document is a high level document which defines the approach for Intelligent Automation testing. It is basically derived from the <Company> Business Process Requirement document. The QA lead for the planned automation should prepare the Test Strategy after consultations with the Manager COE or other Customer Advocate. Where possible the QA Lead should also meet with the Business SME to understand critical business expectations that the automation is being undertaken to address. The Test Strategy sets the standards for testing and is not updated often.

## Test Plan

A Test plan is derived from <Company> Business Process requirement and is prepared by QA or Testing lead for each project. The main goal of test plan is to include all the details related to testing such as what to test, when to test, how to test and who will be the tester. Test plan is not updated often but if there is some new feature or change is introduced then it has to be updated accordingly. Test plan provides guidance to Testing team on all testing that will be performed for each project.

## Test Case

A Test Case is a set of conditions or variables under which a tester will determine whether a system under test satisfies <Company> Business Process Requirements or works correctly. Very importantly, the process of developing test cases can also help find gaps or problems in the Business Process Requirements or design of a Bot. A typical process automation will require multiple test cases to provide coverage for all the functionality and non-functional elements that need to be validated and verified before the Bot development team can sign-off or complete Bot development.

## Defect Management

Defect management is the process of improving quality and software productivity by preventing the injection of defects into the Bot Development Cycle. Although it is virtually impossible to eliminate defects altogether, it is important to have an easily followed defect management approach to ensure timely detection, analysis and resolution to maintain and improve the quality of Bots in development and production. The defect management approach covers defect identification, defect analysis, defect prevention, defect resolution, defect monitoring, and defect process improvement.

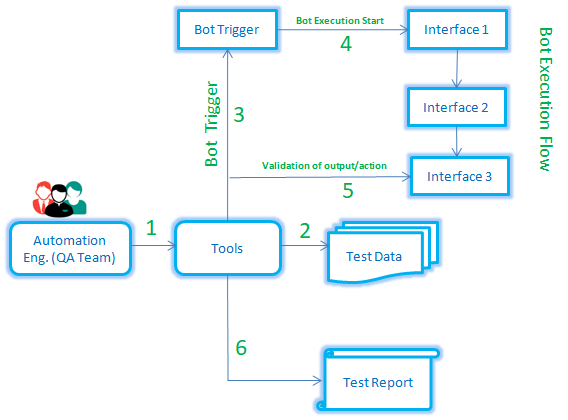
## Traceability

Traceability is the ability to trace relevant artefacts in the development process and **link defects to requirements.** This allows your Bot development team to quickly see which requirements are most buggy. Traceability accelerates and improves development activities. More complete traceability helps avoid software defects and promotes reuse which in turn speeds up development and makes design more modular which are critical ingredients in scaling and maintain large enterprise scale RPA deployments.

Above elements should be integrated into <Company>’s testing framework and test management tools to ensure that complete end to end linkages are available and accessible to both Bot development and other <Company> IT and Software Quality stakeholders.

# Intelligent Automation Testing Approach

Figure below illustrates steps in testing each developed automation. It is assumed that the testing is done using test automation scripting, although the same can also be achieved by manual testing if desired.



**Precondition Check & Test Data Creation**

1. Each Automation script is triggered by the QA team.

2. Automation script retrieves the Test Data.

**Bot Execution**

3. Automation script triggers the Bot through a schedule or from the central UI (Management Console)

4. Intelligent Automation process is initiated and actions are performed by the sequence defined in the Intelligent Automation Bot.

**QA/Testing**

5. Bot automation output is validated to check if Bot actions/output are successful.

6. Intelligent automation creates a Test Report for review by QA Team.

# Testing Environment

Setting up an appropriate test environment ensures Bot testing success. A testing environment is a setup of software, hardware and network configuration to support testing for <Company> intelligent automation Bots. For test environment, key setup requirements include:

* End user/targeted system and applications
* Test data
* Database server
* Front end running environment, etc.
* Bots/RPA platform and deployment in planned production topology

It is recommended to have the following environments available to ensure overall quality practices in design, development and deployment before Bots are deployed to production.

Dev Environment

Here is where the Bot designer develops, tests and checks whether the Bot runs successfully with the design. Once the developed Bot has been tested and the Bot developer feels that the Bot is working fine, the Bot then moves to the staging server. In Dev environment, developers will conduct unit tests which will allow them to conduct basic tests on deliverables prior to completing each sprint.

QA Environment

This environment is ‘owned’ by the QA team. Once Bots are designed they are moved to this environment for full-fledged testing carried out by testers. The testing team performs different types of testing on this environment for developed Bot. Defects found in QA environment, are logged in the defect management system. This type of testing on the QA environment is the final step before the developed Bot could be deployed on a production like environment. QA needs to sign-off before Bots can be deployed in the Staging/UAT environment.

Staging and UAT Environment

This environment is made to look exactly like the production server environment. The Bot is tested on the staging environment to check for reliability, user sign-off and to make sure it does not fail on the actual production server. This type of testing on the staging server is the final step before the Bot could be deployed on a production environment. The developed Bot needs to be approved by users and QA team in Staging/UAT before it can be deployed in the production environment.

Production Environment

The production environment is where the actual functioning Bot with all required applications, systems and other required resources is deployed. The production environment is managed by the Intelligent Automation/RPA Application Owner. QA Team is involved as required when incidents or problems occur and to monitor the performance of the Bot infrastructure in the production environment. However, as per standard IT practice no development or QA work is allowed in the production environment.

# Types of Testing

Following are different types of testing which may need to be conducted during testing cycle. As per Test Strategy and Test Plan for each process, the QA team will develop and conduct the relevant types of tests on each developed automation.

Tests are required on both the Bots/Automated processes and the overall Intelligent Automation Infrastructure that is required for operating all the Bots. The tests to be conducted should be chosen during Test Strategy & Test Plan preparation in consultation with the COE an Bot Development teams and with inputs from concerned <Company> IT stakeholders as required.

Functional Testing

In functional testing all the functional specifications which form the basis for all the tests to be conducted as per <Company> Business requirements are conducted. These requirements and required outcomes are documented in Test Cases as described earlier.

**Emulates Human Action:** Emulates human execution of the repetitive process using various application and systems.

**Perform Multiple Tasks:** Performs multiple and complex tasks across multiple systems. This helps to simulate volumes and complexity that may be encountered during real-life processing of transactions, manipulation of data, preparation of reports, etc.

**Automated report generation:** Automates the extraction of data to come up with accurate, effective and timely reports.

**Information validation and auditing:** Resolution and cross-verification of data between different systems to validate and check information used in and prepared by Bots to provide compliance and auditing outputs.

## Integration Testing

Integration testing is done to test that Bots, components and system interfaces are well integrated to verify that they work as expected during Bot execution. The automated Bots can transfer data between disparate and legacy systems of the <Company>, by connecting them at the user interface level instead of developing new data infrastructure. It is required to test Bot integration between one Bot to other Bot as well as overall integration of the Automation script into <Company>’s IT infrastructure.

**For example**

1. A developed Bot communicates with other Bots as part of integration to accomplish expected outcome.
2. Bot interacts with other set of systems to fulfil expected Business process outcome.

Regression Testing

Regression testing may be required to ensure that any new Bot related operations or upgrades to Bots do not cause any inadvertent impacts on Bots or the execution of automated business processes. As RPA mostly works on the basis of UI automation, there are no code level changes that are usually done. However, advanced types of automation may use APIs, DLLs, other scripting which may require sufficient regression verification.

**For example**

1. A developed Bot required the changes due to Business Process or Infra changes.
2. Application on which Bot is deployed has been upgraded/changed thereby causing failure of Bot and requiring upgrade of Bot.

## Non Functional Testing

**High-volume processing:** Volume testing is a non-functional Performance Test, where the Bot is expected to process a large volume of process data. High volume testing is done to analyze performance of the Bot/Automated Process by increasing the volume of data.

Intelligent automation can easily simulate rekeying of data from one system to another. It performs tasks like data entry, copying, and pasting which lie at the heart of the need for RPA and these must be reliable and repeatable for large volumes of data.

**For example**

1. A developed Bot is reading information b referring to a file or database which contains high volume of data.

**Bot Load Testing:** Each Bot/Automated process should also be tested for load scenarios especially if the Bot is expected to process large volumes of transactions. This is in addition to load testing the overall Bot infrastructure for large numbers of bots. Verifying the performance of the Bot under large volume scenarios helps users and business understand scenarios which can cause its performance to degrade thereby helping in better design, additional work-resource allocation and other measures to be taken to meet business needs.

**Bot Reliability:** Bot should be executed either infinite or finite long duration to ensure it produces expected outcome as per the <Company> business requirements. Reliability metrics should be carefully tracked in testing reports as this is a key source of errors in Bots.

## Performance Testing, Load Testing & Stress Testing

Above types of tests are typically carried out on the Bot platform and infrastructure to determine key performance attributes and resilience of the Bot infrastructure.

Performance testing is conducted to establish the standard behaviour of the RPA infrastructure with the numbers of Bots that will typically operate in production. It should be conducted periodically and updated as the number of Bots/automated processes grow within <Company>’s Intelligent Automation deployment. Measures should be tracked for speed, response time, throughput, resource usage, and stability.

Load Testing tests the system through constantly and steadily increasing the load/number of concurrent Bots on the RPA infrastructure until it reaches the threshold limit. It is a subset of performance testing. Load testing will help identify defects in Bots & infrastructure related to buffer overflow, memory leaks and mismanagement of memory. The issues that would eventually come out as the result of load testing may include load balancing problems, bandwidth issues, the capacity of the existing system, upper limit of all the components of application like a database, hardware, network, etc. so that the application can manage the anticipated load in future. It helps to set the SLAs for the Bot infrastructure.

Stress Testing is designed to overload the available Bot infrastructure with large volumes of jobs carried out in an attempt to shut the Bot infrastructure. Also known as **fatigue testing**, this testing should capture the stability of the Bots and the Bot platform by testing it beyond its bandwidth capacity. It helps the QA team understand the behaviour of the RPA infrastructure beyond peak load and normal conditions. The goal of the stress testing is to analyze post-crash reports to define the behaviour of the application after failure. The biggest challenge is to ensure that the system does not compromise the security of sensitive data after the failure. In a

## Security & Vulnerability Testing

Security and Vulnerability Testing is required to be conducted on the Bot platform and specific Bots to meet <Company>’s IT Security and Compliance standards. Security/Vulnerability testing is conducted to reveal any flaws in the security mechanisms and handling/manipulation operations of systems that protect data and maintain functionality as intended while Bot is running. Typical security requirements may include specific elements of confidentiality, integrity, authentication, availability, authorization and non-repudiation. RPA applications provide for Data Security and Encryption for Data at Rest and Data in Motion scenarios. Security tests should verify and confirm the functioning of these security mechanisms.

QA Team should coordinate with <Company>’s IT Security & Compliance team to ensure adequate security & compliance testing is conducted to meet requirements prior to deployment to production. As there are several regulatory and customer specific information security compliance requirements at <Company>, the QA and Solution Design team should take this into consideration from Design stage of projects.

## High Availability (HA) and Disaster Recovery (DR) Testing

High availability testing for Bots ensures that Bots are available 24x7 365 days in a year. This is one of the critical reasons why Bots are put into production by businesses as Bots can then complement the human workforce. Therefore it is essential to ensure Bots and the Bot platform and infrastructure are able to respond to unexpected hardware or software failure of some of its components. High availability testing will provide basis for planning to prevent outright failures of the Bot infrastructure and to ensure continuous operation, therefore increasing fault tolerance and reliability. The goal of High availability testing is not necessarily to prevent faults, but to ensure design of Bots and overall Bot infrastructure for tolerance, recoverability and reduced occurrence of high severity faults. This testing measures business impact from faults and helps in planning for reducing impact. Other elements of High Availability testing will help <Company> COE & QA Team to Understand and design better for points of failure, Monitoring mechanisms and Develop a fault model; all of which will lead to a more resilient and performing Bot infrastructure.

RPA platforms provide for several HA and DR capabilities. As part of the testing cycle for each Bot, if required and periodically for the overall RPA infrastructure, these capabilities of the RPA platform should be tested as per <Company>’s HA & DR drills.

## UAT (User Acceptance Testing)

User acceptance testing (UAT) is the last phase of the software testing process. During UAT for Bots/Automated Processes, actual Bot users (here <Company> Business User/SME) tests/verfies the Bot execution outcome to make sure it can handle required tasks/process in real-world scenarios. UAT is conducted in the pre-production environment hence should be kept ‘clean’ and no changes should be planned in the UAT environment to avoid surprises in the Production environment once the Bot is signed-off by Users and QA Team.

# Test Data

Test Data is created or provisioned by the <Company> Business team, Business SME. It is strongly recommended to mimic the production data and apply high levels of control over test data to maintain its quality. In each testing scenario, each Bot/Intelligent Automation script for <Company> business process will use the required test data during execution. Sufficient and relevant test data needs to be available for used throughout testing cycle for Bot execution in different environments. Intelligent Automation QA Team should oversee the maintenance of the test data on ongoing basis.

# QA Sign-Off Report

At the end of each QA cycle, QA team will produce the QA sign-off report. This report will contain the execution summary, defect summary and result of each type of testing. The QA Sign-off report and UAT forms the basis for release of the Bot into production.

QA sign-off report will help all stakeholders to stay in sync about the health of the developed Bot. It will also help to understand Bot execution impact on interface, infrastructure and business process.

# Testing Tools

Intelligent Automation COE’s QA Team should aim for a healthy mix of manual and automated testing. As a matter of good QA & Testing practice, increasing Automated Test Coverage is an important metric to target. It is recommended that from a tools perspective the Intelligent Automation QA Team should align with standard <Company> QA tools for defect tracking, traceability, test case management and test automation.

# QA & Testing for other Intelligent Automation Technologies

While this document focuses on QA & Testing for RPA enabled automation deployments, it should be kept in mind that Intelligent Automation deployments may involve multiple other technologies including AI, ML, OCR, ChatBots or other specialized technology components. The QA Team in their planning should ensure that in cases involving any other such automation technologies, the QA team should have sufficient technical and functional understanding of these technologies so as to plan for appropriate testing.

# Appendix A: Defect Log template

The defect log will help all team members to study and understand sufficient details about the defect found during QA cycle. It will also help to track towards to close the defect. It should be available and tracked in a defect management tools like JIRA, TFS, etc (as per <Company> standard tools).

Standard details to be entered by QA:

* Defect ID (Unique)
* Version
* Platform details
* Application/Interface
* Summary/Title
* Description
* Expected Behaviour
* Actual Behaviour
* Severity
* Priority (Must be entered by Business Owner)
* Supported Log, Snapshot

# Appendix B: QA Test Strategy

Following are the standard details which are required as part of a QA Test Strategy.

* Scope and objective: The objective of the business and how much testing scope is there is defined under test strategy.
* Business Issues: How much is the budget of the project, how much time is required for testing, how much resources are needed etc. are the part of business issues which needs to be considered before the actual testing starts.
* Testing approach: What type of testing is needed (performance, load, stress, functional etc.) and whether the testing is only manual or automation or both are some of the crucial points which define the testing approach.
* Test deliverables: What are the documents required from the testing team, how they would keep the record of the testing cycles etc. will be included here.
* Defect tracking approach: Which tool will be used for tracking the defects and how will the testing team communicate with the development team and how the flow would go for defects are decided at this point in test strategy.
* Training: If there is some complex or new tool is introduced in the business then it is helpful if the team members are given proper training. What type of training and the responsible person to conduct such training is defined here?
* Automation: If the project or business needs automation testing then the script language, tool used, reporting and code maintained is planned in test strategy.
* Risks: Nobody can anticipate all the risks beforehand but obvious risks can be avoided and also solution (if risk occurs) can be included in the document for future help.

# Appendix C: QA Test Plan

Following are the standard details which are required as part of a QA Test Plan.

* Test plan ID: This is a unique ID which defines the test plan. It can be a number or name or mix of both, as per the convenience.
* Test environment: This section defines what kind of environment is needed for the testing to carry out. For e.g. in device testing, usually a virtual set up is made to test emergency calling.
* Features to be tested/Not tested: This will have all the details about the features which tester needs to test and what are the feature which are not tested (may be because it is not yet implemented or not tested for that particular release).
* Entry/Exit criteria: These are the terms which define when to start or stop the testing. Standards will be defined under test strategy and followed by testers in test plan.
* Status: Whether a test case is passed or failed or not tested, all these test results are included in test plan with a proper reason.
* Types of testing: The types of testing required such as regression, functional, non-functional, stress etc. are defined and then executed by the respective tester.
* Brief Intro: Brief introduction is also included sometimes so that if any new member joins the team, he should get an idea how things work.