

Apprity-GitHub Connector Test Plan

Version: Draft



This document contains privileged and confidential information which is the property of Persistent Systems Ltd.

Document History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Description** | **Author** | **Reviewed By** | **Approved By** | **Release Date** |
| Draft v1.0 | Created Test Plan 1st Draft version. | Sunil Singh |  |  | 29th Sep 2014 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Contents

Contents

[Document History 1](#_Toc399876340)

[2 References 3](#_Toc399876341)

[3 Introduction 3](#_Toc399876342)

[3.1 Scope 3](#_Toc399876343)

[4 Testing Strategy 5](#_Toc399876344)

[4.1 Test Objectives 5](#_Toc399876345)

[4.2 Test Environment 5](#_Toc399876346)

[4.3 Test Approach: 6](#_Toc399876347)

[4.3.1 Test Cycles or execution 6](#_Toc399876348)

[4.3.2 Functional Testing at each release Level 6](#_Toc399876349)

[4.3.3 System Integration Testing 7](#_Toc399876350)

[4.3.4 UAT 7](#_Toc399876351)

[4.4 Performance measurement 7](#_Toc399876352)

[4.4.1 Approach 7](#_Toc399876353)

[4.4.2 Exclusion 8](#_Toc399876354)

[4.5 Test data 8](#_Toc399876355)

[4.6 Testing Techniques 8](#_Toc399876356)

[4.7 Test Areas 8](#_Toc399876357)

[4.7.1 Functionality 8](#_Toc399876358)

[4.8 Error scenarios/Negative testing 10](#_Toc399876359)

[4.9 Deployment Steps 10](#_Toc399876360)

[4.10 Build Verification testing 10](#_Toc399876361)

[5 Test Case and Strategy 10](#_Toc399876362)

[5.1 Test coverage 10](#_Toc399876363)

[5.2 Acceptance test scenarios 10](#_Toc399876364)

[5.3 Testing with different Browsers 15](#_Toc399876365)

[5.4 Functional Limitations 15](#_Toc399876366)

[5.5 Security 15](#_Toc399876367)

[5.6 Test Case Management Overview 15](#_Toc399876368)

[5.6.1 Test case type as ‘Category’: 15](#_Toc399876369)

[5.6.2 Test case status 16](#_Toc399876370)

[5.6.3 Test case priority 16](#_Toc399876371)

[5.7 Completion Criteria (Suspension or Exit criteria) 16](#_Toc399876372)

[5.7.1 Entry Criteria: 16](#_Toc399876373)

[5.7.2 Exit Criteria: 17](#_Toc399876374)

[6 Test Automation 17](#_Toc399876375)

[7 Out-of-scope 17](#_Toc399876376)

[8 Testing Milestones 17](#_Toc399876377)

[9 Test Execution Deliverables 18](#_Toc399876378)

[10 Defect Management 19](#_Toc399876379)

[10.1 Defect Logging 19](#_Toc399876380)

[10.1.1 Guidelines 19](#_Toc399876381)

[10.1.2 Blocking 19](#_Toc399876382)

[10.1.3 Severity 19](#_Toc399876383)

[10.1.4 Priority 20](#_Toc399876384)

[10.1.5 Defect Resolution 20](#_Toc399876385)

[11 Test reporting 20](#_Toc399876386)

[11.1 Test Logs 20](#_Toc399876387)

[11.2 Defect Reports 20](#_Toc399876388)

[11.3 Test Summary Reports 21](#_Toc399876389)

[12 Assumptions and constraints 21](#_Toc399876390)

[13 Risks and mitigation 22](#_Toc399876391)

[14 Sign-off 22](#_Toc399876392)

# References

1. Persistent - Apprity SOW September 22nd, 2014.
2. Software Requirements Specifications-Apprity-GitHub connector.

# Introduction

## Scope

This document covers the test plan for end to end testing of Apprity GitHub connector. The Apprity-GitHub connector will enable Apprity to crawl data from GitHub.com cloud source.

Following are the major operations that Apprity connector is going to support,

* Apprity-GitHub connector will connect to GitHub.com using token based authentication for using the GitHub API and User Name, password for web Scraping.
* Connector should support crawling and content retrieval for events which are as mentioned below: e.g.
* Team: Events
  + Member of an organization added to team: Add\_member.
  + Team is given control of a repository: Add\_repository.
  + A new team is created: Create.
  + Team from the organization in deleted: Destroy.
  + Member from organization is removed from the team: Remove\_member.
  + Repository is no longer under a team's control: Remove\_repository.
* Repo: Events
  + Repository owned by an organization is switched from "private" to "public" (or vice versa): Access.
  + GitHub user is given collaborator access: Add\_member.
  + New repository created: Create.
  + Repository is deleted: Destroy.
  + Repository is disabled: Disable.
  + Repository is re-enabled: Enable.
  + Remove a user as a collaborator: Remove\_member.
  + Repository is renamed: Rename.
  + Ownership of repository changed Transfer.
  + Before repository transfer occurs: Transfer\_start.
* Webhooks: Events
  + New hook to a repository in the organization: Create.
  + Existing hook has its configuration altered: Config\_changed.
  + Existing hook was removed from a repository Destroy.
  + Events on a hook have been altered: Events\_changed.
* Org: Events
  + New member invited: Invite\_member.
  + Member removed from organization: Remove\_member.
  + Permission of member changed: Update\_member.
* Others: Events
  + Listing teams for an organization.
  + Public members in an organization.
  + All members in an organization.
  + Followers of a user.
* Connector should retrieve Logs/Events for Audit Monitoring of Data Source as configured through web scraping.
* Connector should store all the information-connection, log/events, and configuration in the Cassandra DB. Change/adjust Cassandra schema should be as per the Data source supported attributes.
* The Connector should monitor Cloud GitHub private repositories only. The scope of the connector will not include any write back to GitHub.
* The connector should invoke/ consume the web services exposed by GitHub to fetch relevant information and add it to the Apprity database.
* Apprity UI should extract information’s related to configuration statistical graphs based on data sent by connector for the Data Source.
* Change detection for an organization repository, deleted repository and modified repository should be crawled in the incremental mechanism supported by Apprity GitHub connector.
* Change detection for a new organization info added, info deleted and info modified, should be crawled in the incremental mechanism supported by Apprity GitHub connector.
* Change detection should be supported for User activity, Team, Repo, Webhooks, Org and Others Activity.
* For multiple instance of GitHub Create User, Modify User, Delete User should be supported and its specific information should crawl by Apprity GitHub connector.
* Read only registration i.e. RegisterAppInstance-MonitorOnly should be supported by Apprity GitHub connector. In this case related settings should not be pushed to cloud.

Please note that the document is live and will be getting modified based on the findings from the test results, so as to accommodate for the changes in architecture, test cases, test tools, and the methodology as well.

# Testing Strategy

Once requirement is fixed, testing team will start writing test cases based on the requirements. The low level test cases will include both valid and invalid inputs to test the connector.

## Test Objectives

* Validate connector based on scenarios mentioned in the SRS document.
* Define Test approach and execution methodology.
* Collect performance parameters and measurements.
* Identify the Test deliverables and major milestones of the Test activities.
* Provide the Test report/List of Defects daily to development team and weekly to the management/Apprity.
* Test and raise any bugs / issues / suggestions with the deployment steps (documents / user guides) for the connector.
* To identify and close (resolve) all P0, P1 bugs and issues recorded during the testing.

Note: The actual resolution for all the defects will come from individual designers / developers for the related components.

* To sign-off for the subsequent stage of the project (post testing) depending upon the test results approval.

## Test Environment

* CentOS 6.4 (64-bit).
* Software: Tomcat 7.0 or higher, DSE - 4.5.1, Python 2.7, performance monitoring tools for CentOS equivalent to perfmon.
* GitHub test data for multiple users.
* Access to JIRA for bug tracking.

## Test Approach:

Once requirements are fixed and approved in SRS document testing team will start writing test cases based on the SRS. The test cases will include both valid and invalid scenarios to test the connector.

#### Test Cycles or execution

We have planned to carry out one or multiple test cycles of test case execution in each phase of the project depending on the number of high priority bugs or product stability.

As per the current project plan requirements are divided in to following 3 phases.

|  |  |
| --- | --- |
| Phase | Requirements |
| Preliminary | Apprity-GitHub Connector should be delivered as WAR file and all requirements from the SRS document.  On priorities, connector implementation should include these functionalities: Crawling of User activity, Team activity, Repo activity, Webhooks Activity, Organization activity and Others activity. Incremental crawl should be supported. Correct analytics data with proper representation in UI should be supported. |
| Final | Product documentation + Preliminary deliverables. |
| Revised | Customer feedback + Final deliverables. |

Note: The table covering the requirements below will change as per the new project plan.

Following types of testing will be performed in each phases. Test execution in each cycle may vary depending on the number of test cases, severity of bugs and timelines.

#### Functional Testing at each release Level

Functional testing is a type of black box testing that bases its test cases on the specifications (mentioned in low level design) of the software component under test. Functionality testing will verify whether the connector meets the intended specifications and functional requirements.

Functionality testing will involve following steps:

1. The identification of functions that the software is expected to perform.
2. The creation of input data based on the function's specifications.
3. The determination of output based on the function's specifications.
4. The execution of the test case.
5. The comparison of actual and expected outputs.

#### System Integration Testing

System integration testing (SIT) is a testing process that will exercise system's coexistence with others. System integration test cases will be based on the finalized High Level Design. The document for system integration testing will contain end-end test cases where integration scenarios based on Apprity and GitHub will be considered.

#### UAT

User Acceptance Testing is final step before rolling out the application. This type of testing gives the end users the confidence that the application is being delivered to them meets their requirements. This testing will involve running a suite of tests on the completed system. Each individual test, known as a case, exercises a particular operating condition of the user's environment or feature of the system.

Acceptance test checklist based on the SRS will be shared with Apprity well in advance.

## Performance measurement

The connector will be monitored for data retrieval in ideal conditions for following parameters

* Time.
* CPU usage.
* Memory utilization.

Note: Performance measurement will be executed for the latest stable version of the connector.

#### Approach

* Approximate volume for a full crawl would be 20 repositories with 500 files and 50 users having 10 transactions a day over a period of 180 days.
* Performance will be measured using logs generated by Apprity-GitHub connector. Suitable logging level will capture time duration for the actions performed at Apprity and GitHub.
* We will use any performance monitoring tools for CentOS equivalent to perfmon.

#### Exclusion

* Performance of connector may deviate depending on the network bandwidth set up.
* Analytics performance monitoring is out of scope.

## Test data

Following test data will be considered. This list may be modified later.

1. Persistent test team will create test data as required on test machine based on test cases.
2. Volume test data having different variety to be created for performance testing.

## Testing Techniques

Once requirements are finalized, testing team will start writing test cases based on the requirements. The test cases will include both valid and invalid inputs to test the system.

1. Equivalence Partitioning
2. Boundary Value Analysis

## Test Areas

#### Functionality

In order for Apprity to be able to monitor activities on GitHub, there is a need for a connector between Apprity and GitHub. The connector will invoke/ consume the web services exposed by GitHub to fetch relevant information and add it to the Apprity database. Since the GitHub API does not provide much information needed for monitoring we will use web scraping to extract information for the admin users audit log page.

Listed below is the detailed functional requirement for the connector.

1. Connector would be used to fetch information for organization private repositories on GitHub.
2. To be able to have full access to organization information on GitHub, the connector will use an administrator account.
3. Connector will be using token based authentication to connect to GitHub using the GitHub APIs, whereas for the web scraping, basic authentication will be used.
4. The connector will read the GitHub feeds, events using a combination of APIs and web scraping to get following categories of event
   1. User activity
   2. Organization activity
   3. Repository activity: access type, collaborators, Addition & deletion of repositories, transfer of ownership
   4. Team activity : members add & delete, Repository add/ deletes
   5. Webhooks: Creation, Updates and deletion
5. Since GitHub does not provide for setting any policies, policy monitoring will not be in scope
6. Data fetched will be written to the events tracking table in the Apprity database.
7. GitHub does not provide incremental event logs. The alternatives we will use are
   1. In case of web scraping, we can create a query string with filters for created date and append to the URL (e.g. [https://github.com/orgs/OrgName/audit-log?q=created%3A>%3D2014-07-08](https://github.com/orgs/OrgName/audit-log?q=created%3A%3e%3D2014-07-08) for events created after 2014-07-08) and get the required subset of data. Since the created parameter does not include the time we will still have to add some bit of filtering from our end so as to ensure we don’t have duplicate events or miss out on any events on a particular day of query.
   2. For the API (for events & feeds) there is no way to get it incremental information Solution is to maintain at timestamp of last read events and add events added after the last read
8. Connector will be invoked through the Apprity scheduler.
9. Apprity UI will be updated to allow user to add GitHub as a source application
10. Data/event retrieve from GitHub will be mapped to threats, incidents/risks and relevant KSIs will be generated. This would require modification to the analytical algorithms that Apprity uses.
11. The current Apprity UI will also be updated to support representation of the analytics for the data retrieved from GitHub. This involves two parts,
    1. Publishing or modifying existing REST APIs that will retrieve data from Cassandra
    2. Changes to Apprity UI component to consume these REST APIs

**Note**: Above functionality may be changed as per the latest SOW/SRS.

## Error scenarios/Negative testing

The goal of error tests is to simulate error scenarios and identify the system behavior upon component failures to make sure the system responds appropriately and restores gracefully.

To include –

* Network failures.
* System / Component failures.
* Unsupported file type or size.

## Deployment Steps

The goal is to validate the installation and configuration steps, mentioned in deployment and administration guide in testing environment.

## Build Verification testing

The goal of this sanity test is to ensure that build is stable and suitable for acceptance into system testing. In case provided build is not stable then build will be rejected in Sanity testing with proper step to reproduce the defect. In such cases detailed steps will be provided to the development team.

Note: The BV testing would include positive test cases along with test cases reported as failures during the last run of system testing.

# Test Case and Strategy

Test cases will be entered manually in excel sheet for test case management.

## Test coverage

Testing team will focus Apprity connector verification on CentOS version mentioned in the SRS and GitHub.com

## Acceptance test scenarios

Note: Testing will validate the functionality with different preconditions with respect to requirements for all the use cases mentioned below.

|  |  |
| --- | --- |
| Acceptance: ID | Description |
| Apprity\_GitHub \_01 | Apprity-GitHub connector should be delivered as a WAR file. |
| Apprity\_GitHub \_02 | The Connector should be compatible with GitHub. |
| Apprity\_GitHub \_03 | Check for user registration with valid input data. |
| Apprity\_GitHub \_04 | There should be a provision to enter the account name and password that should be used to connect to GitHub to retrieve data in connector configuration.  Note: Use an administrator account. |
| Apprity\_GitHub \_05 | There should be a provision to configure the account name and password that will be used to connect to GitHub to retrieve data in connector configuration.  Note: Use a simple account |
| Apprity\_GitHub \_06 | Check for password which gets saved in Apprity database should be encrypted. |
| Apprity\_GitHub \_07 | Connector should be invoked through the Apprity scheduler. |
| Apprity\_GitHub \_08 | Connector should fetch information for organization private repositories on GitHub.  Note: Use an administrator account. |
| Apprity\_GitHub \_09 | Using connector, an administrator user should be able to have full access to organization info on GitHub. |
| Apprity\_GitHub \_10 | Connector should use token based authentication to connect to GitHub using the GitHub APIs, |
| Apprity\_GitHub \_11 | Connector should use basic authentication for the web scraping. |
| Apprity\_GitHub \_12 | Connector should support crawling and content retrieval across User activity as configured. |
| Apprity\_GitHub \_13 | Connector should use Web scraping to monitor “invite\_member” for an organization. |
| Apprity\_GitHub \_14 | Connector should use Web scraping to monitor “remove\_member” for an organization. |
| Apprity\_GitHub \_15 | Connector should use Web scraping to monitor “update\_member” for an organization. |
| Apprity\_GitHub \_16 | Connector should use Web scraping to monitor “access” level on repository i.e. Repository owned by an organization is switched from "private" to "public" (or vice versa). |
| Apprity\_GitHub \_17 | Connector should use API to monitor “add\_member” in the repository. In this case GitHub user is given collaborator access. |
| Apprity\_GitHub \_18 | Connector should use API to monitor “create a new repository”. |
| Apprity\_GitHub \_19 | Connector should use Web scraping to monitor “destroy an existing repository”. |
| Apprity\_GitHub \_20 | Connector should use Web scraping to monitor “disable a repository”. |
| Apprity\_GitHub \_21 | Connector should use Web scraping to monitor “enable a repository”. |
| Apprity\_GitHub \_22 | Connector should use Web scraping to monitor “remove\_member from existing repository”. |
| Apprity\_GitHub \_23 | Connector should use Web scraping to monitor rename an existing repository. |
| Apprity\_GitHub \_24 | Connector should use Web scraping to monitor transfer a repository e.g: ”Ownership of repository is changed changed” |
| Apprity\_GitHub \_25 | Connector should use Web scraping to monitor start transfer “transfer\_start: Before repository transfer occurs” |
| Apprity\_GitHub \_26 | Connector should use API to monitor add a new member i.e. add\_member: Member of an organization added to team. |
| Apprity\_GitHub \_27 | Connector should use API to monitor when Team is given control of a repository. |
| Apprity\_GitHub \_28 | Connector should use API to monitor create a new Team. |
| Apprity\_GitHub \_29 | Connector should use Web scraping to monitor destroy i.e. Team from the organization in deleted. |
| Apprity\_GitHub \_30 | Connector should use Web scraping to monitor remove\_member i.e. Member from organization is removed from the team |
| Apprity\_GitHub \_31 | Connector should use Web scraping to monitor remove\_repository i.e. Repository is no longer under a team's control. |
| Apprity\_GitHub \_32 | Connector should use Web scraping to monitor create webhooks i.e. New hook to a repository in the organization |
| Apprity\_GitHub \_33 | Connector should use Web scraping to monitor config\_changed webhooks i.e. Existing hook has its configuration altered. |
| Apprity\_GitHub \_34 | Connector should use Web scraping to monitor destroy webhooks i.e. existing hook was removed from a repository |
| Apprity\_GitHub \_35 | Connector should use Web scraping to monitor events\_changed webhooks i.e. Events on a hook have been altered. |
| Apprity\_GitHub \_36 | Connector should use API to monitor “Listing teams for an organization”. |
| Apprity\_GitHub \_37 | Connector should use API to monitor “Public members in an organization”. |
| Apprity\_GitHub \_38 | Connector should use API to monitor “All members in an organization”. |
| Apprity\_GitHub \_39 | Connector should use API to monitor “Followers of a user”. |
| Apprity\_GitHub \_40 | Verify that list of monitoring parameters can change subject to changes in the audit log actives maintained by GitHub.com |
| Apprity\_GitHub \_41 | Verify that the events should be mapped to the Apprity model. |
| Apprity\_GitHub \_42 | Since GitHub does not provide for setting any policies, policy monitoring will not be in scope |
| Apprity\_GitHub \_43 | Connector should be using GitHub API v3, jdk v1.7.0\_02 |
| Apprity\_GitHub \_44 | Data fetched should be written to the events tracking table in the Apprity database. |
| Apprity\_GitHub \_45 | Connector should support Full and Incremental crawl. |
| Apprity\_GitHub \_46 | Connector should support incremental event logs for API as:  For the API (for events & feeds) there is no way to get it incremental information Solution is to maintain at timestamp of last read events and add events added after the last read. |
| Apprity\_GitHub \_47 | Apprity UI should be updated to allow user to add GitHub as a source application. |
| Apprity\_GitHub \_48 | Data/event retrieve from GitHub should be mapped to threats, incidents/risks and relevant KSIs will be generated. This would require modification to the analytical algorithms that Apprity uses. |
| Apprity\_GitHub \_49 | The current Apprity UI should also be updated to support representation of the analytics for the data retrieved from GitHub. This involves two parts,  a. Publishing or modifying existing REST APIs that will retrieve data from Cassandra  b. Changes to Apprity UI component to consume these REST APIs |
| Apprity\_GitHub \_50 | The Connector should make available new and updated entities to Apprity. |
| Apprity\_GitHub \_51 | The Connector should indicate removal of all such entities/ documents from the index that have been deleted from source. (Need to confirm, if index not required will remove it) |
| Apprity\_GitHub \_52 | The Connector should record its transactions into a log file |
| Apprity\_GitHub \_53 | A user guide documenting configuration and operation of the Connector using Apprity templates and guidelines should be provided. |

## Testing with different Browsers

* All test cases would be executed with IE Version X.
* Smoke testing on Mozilla and Chrome browsers.

## Functional Limitations

GitHub does not provide API’s for the following:

* Network / session monitoring. There is no way for organization admin to get user’s login information.
* Password policy – there is no way to set or monitor password policies.
* Download of Audit logs.

## Security

Typical security requirements for Apprity-GitHub connector should include specific elements of confidentiality i.e. password should be saved in encrypted type.

## Test Case Management Overview

All test cases will be associated with the Requirement ID (as per SRS document). Low level test design will include Test ID, Req No, Title, Date,

Version, Category, Priority, Test Pre-Requisites, Steps To Execute Test, Expected Results to Verify Test, Status, Bugs and Comments.

#### Test case type as ‘Category’:

* Functional
* Smoke
* Performance
* Security

#### Test case status

|  |  |
| --- | --- |
| Option | Description |
| Idle | Mark as Idle not run. |
| Pass | Mark as Passed. |
| Fail | Mark as Failed. |
| Running | Mark as Running. |
| Pause | Mark as Paused. A temporarily test case (due to pending decisions on system behavior). Can be re-activated or retired at a later date. |
| Block | Test case marked as Blocked due to something not being implemented yet or a dependency requirement not working. |
| Error | Test case does not seem to be valid with respect to Requirement. May need to be Removed at a later date. |

#### Test case priority

* Priority-1\*: Critical tests: Allocated to the tests, which must pass, otherwise the sign -off will be affected. Target very basic tests, e.g. installation\setup\configuration or Apprity connector connectivity with GitHub etc. In another word it is a kind of smoke test.
* Priority-2: High priority tests: Allocated to the tests, which must be executed before the final delivery. Common functionality that is used by most user’s: Connector should crawl and retrieve entities e.g. Team, Repo, Webhooks, org and others. Incremental crawl and logging log messages at information level etc.
* Priority-3: Medium priority tests: Allocated to the tests which can be executed, once Priority1 and Priority2 test cases are executed completely.
* Priority-4: Low priority tests: Allocated to the tests, which can wait & can be executed which are related to log messages at debug and higher levels.

## Completion Criteria (Suspension or Exit criteria)

#### Entry Criteria:

Testing team assumes that product management team and the development team have finished with the design phase for the connector with Test harness utility. As a result of this process, they have accomplished following tasks:

1. Release scope item list is locked and prioritized.

2. Functional specification or acceptance testing checklist document for the connector is completed and signed off.

3. Development team has finished implementing all the features accepted during project plan.

4. Development team has finished unit testing for all the modules

#### Exit Criteria:

1. All written test cases are executed and all P1 and P2 test cases pass.

2. There are no open P0 and P1 defects in the application.

# Test Automation

* Automation will not be considered for the connector testing.

# Out-of-scope

* No write back to GitHub is considered.
* Test Automation is out of scope.
* GitHub does not provide incremental event logs hence will not be implemented.

# Testing Milestones

Following will be the Milestones:

|  |  |  |  |
| --- | --- | --- | --- |
| Milestone | Date | Deliverables | Owner |
| Test Plan (This document) | 09/30/2014 | Test Plan document | Sunil Singh |
| Preview 1 delivery |  | Test execution document, Defects report | Sunil Singh |
| Preview 2 delivery |  | Test execution document, Defects report | Sunil Singh |
| Beta connector delivery |  | Test execution document, Defects report, Release notes | Sunil Singh |
| Final delivery acceptance |  | Test execution document, Defects report, Release notes, Updated Acceptance checklist | Sunil Singh |

# Test Execution Deliverables

The following will be the deliverables by the test Team for the connector testing.

|  |  |
| --- | --- |
| Deliverable | Responsibility of |
| Test Plan. The Test Plan breaks functionality into logical areas (most often specified in the SOW Document). Once reviewed and amended, it must be approved and signed by all interested parties. | QA |
| Test Cases. Each test case includes the steps necessary to perform the test, expected results and contains (or refers to) any data needed to perform the test and to verify that it works. | QA |
| Final Test Summary Report. This report will identify the % complete of all tasks that should be due by that milestone, the tasks to be worked on in the next milestone, bug and test case metrics (once testing begins), and any issues or risks that need to be addressed. | QA |
| Release Notes with every delivery of Apprity GitHub Connector. It will include release features, known defects or limitations. | QA |
| Acceptance Checklist Test case from acceptance checklist will be identified as UAT test cases and communicated to the client as part of deliverables. | QA |

# Defect Management

## Defect Logging

#### Guidelines

* Enter the exact reproducibility steps, where ever possible.
* All details to be captured and communicated through the bug logged.
* If required activity log file to be attached with respective bug ID.

#### Blocking

If the bug significantly blocks testing then the bug should be marked as “Blocker\_QA”. It is important to capture this information since the testing track gets closed due to the blocking. These bugs would significantly impact the test schedule.

Once a bug has been marked as Blocker\_QA – this value must not be removed unless it was added in error.

A bug is “Blocker\_QA” if:

* It prevents further execution of test cases around a particular component or feature area.
* It prevents deployment of the solution or a feature of the solution.
* In the case of documentation, the instructions are so incorrect as to be impossible to follow or workaround safely without potentially harming a deployment. This is particularly important during the final integration test pass.

#### Severity

Severity of the bug will be assigned as per the table below:

|  |  |
| --- | --- |
| Severity | Description |
| Blocker\_QA | Fatal error, crash, blocking issue. |
| Critical | Non-fatal error, still some major functionality has failed. |
| Major | Part of the functionality failed. |
| Minor | UI, customization, configuration related issues. |

#### 

#### Priority

Priority of the bugs will be assigned during the bug review meetings.

|  |  |
| --- | --- |
| Priority | Description |
| P0\* – Critical | Either all or more than one requirement are affected and it is MUST to fix them (need a patch/new build ASAP, can’t wait for next build). |
| P1 – High | One of the major functionality/requirement is affected and MUST to fix them. |
| P2 – Medium | Part of functionality/requirement is affected (there is possible work around). |
| P3 – Low | Nice to have. |
| P4– Enhancement | Any enhancement or suggestions based on usability. |

#### Defect Resolution

The defects would be fixed as per the priorities identified. Build with the fixed defects will be installed on test environment. Testing team will verify the fixed defects.

# Test reporting

## Test Logs

Results will be logged on an ongoing basis as test cases are executed to ensure up to date reporting.

Test results will be entered manually after completion of each test run (as all tests in phases are manual) in an excel sheet which will include Test ID, Req No, Title, Date, Version, Category, Priority, Test Pre-Requisites, Steps To Execute Test, Expected Results to Verify Test, Status, Bugs, Comments.

## Defect Reports

JIRA will be used for Defect Tracking.

Detailed steps will be updated in this doc. Please refer below snapshot for more details.

TBD

## Test Summary Reports

This report will contain the test execution log file in .xls format including completion report based on individual Test Run cycle.

# Assumptions and constraints

* Connector will monitor Cloud GitHub private repositories
* GitHub provides APIs necessary for monitoring activities listed earlier
* Connector will be using GitHub API v3, JDK v1.7.0\_02, jsoup 1.7.3, DataStax 4.5.1.
* No write back to GitHub is considered.
* Apprity will confirm on the list of activities to be monitored.
* Members from current Apprity team at Persistent will help from Apprity integration perspective.
* Unknowns, if any that would affect the timelines would be addressed in version 2.0.
* Test Automation is out of scope.
* Performance testing to be done only for data retrieval from GitHub.
* Connector performance and scalability will be achieved on best-effort basis. Any specific performance tuning requirements will be out of scope for this engagement.
* Limitations
  1. GitHub does not provide APIs for the following:
     + Network / session monitoring. There is no way for organization admin to get user’s login information
     + Password policy – there is no way to set or monitor password policies
     + Download of Audit logs
  2. The audit logs page does not log the time an event occurred. It displays only the date when the event occurred. This could be restricting on the analytics pattern matching.

# Risks and mitigation

Following are the identified risk and mitigation plan that may impact testing schedule and execution.

|  |  |
| --- | --- |
| **Risk** | **Mitigation** |
| Requirements: The test plan and test schedule are based on the current SRS Document. Any changes to the SRS Document could affect the test schedule. | Additional effort for testing the changed or added new requirement will be communicated to the client. |
| Scope of testing: After the first round of testing, subsequent cycles will focus mostly on high priority test cases and those test cases which have defects against them. | Test cases will be shared with client and the entire project team as contingency steps. Reproducing the bug in QA environment and analyzing the impact of the bug on product. |
| Testing schedule may get affected if review and signoff is been delayed from Apprity. | The delay will be communicated with the client and if schedule disturbed no. of regression test cases may be shirked. |
| Performance testing: Performance testing will be performed in PSL environment as specified in the Test Environment section above. | QA will try to have performance testing environment as close as possible to production environment. Configuration/scenarios will be communicated with Apprity. |

# 

# Sign-off

This document will be signed off by the following stakeholders:

1. Ganesh Kirti: CTO, Apprity.
2. Shashank Deshpande: Project Manager, Persistent Systems Ltd.