Neptune DXP 24 - Open Edition

Revision History

Date	Version	Author	Description
18.05.2022	1.0.22	Rommel Lamanilao	Test plan for Neptune DXP Open Edition 22-LTS release
28.09.2023	1.0.23	Rommel Lamanilao	Test plan for Neptune DXP Open Edition 23-LTS release
10.09.2024	1.0.24	Rommel Lamanilao	Test plan for Neptune DXP Open Edition 24-LTS release

Table of Contents

1. INTRODUCTION

1.1. Overview

Neptune DXP — LTS (long-term support) version offers all the features rolled out in one production-ready release. Prior to its actual release, several tests must be executed to ensure that at least all major functionalities are working. This test plan is created to guide the QA Team for the entire testing process.

Also, this document is intended for the testing process of **Neptune DXP — Open Edition 24-LTS** version.

2. OBJECTIVES AND TASKS

2.1. Objectives

The person in charge of the tests should pay attention to this objective before actual tests are performed. The initial release notes can be used as reference to help meet these objectives.

The objectives of the test should be:

- that all new features are working as expected
- that all bug fixes should be tested and verified
- that all major and minor functionalities should be working
- that all other tests defined should PASS

2.2. Tasks

The activities involve in this test plan are the following:

- Determine the scope of the test — tests to be performed and NOT to be performed
- Document the Test Strategy
- Decide the test criteria — suspension and exit criteria
- Plan the test resources
- Plan the test environment
- Plan when and how to test
- Deliver test results as part of the execution

3. SCOPE AND LIMITATIONS

3.1. Scope

Testing will mostly cover the following:

3.1.1. New Features and Enhancements

More details on how to test these can be found in the Test Management tool.

- **Naia**
  - List of applicable missions in the cockpit app header
  - New Naia cockpit tile showing available missions
  - Naia is now available as a side panel
- **Cockpit**
  - App icon now works like the “close” button
  - New look user settings
  - New option to display navigation menu in the side panel
  - Cockpit apps can now be toggled as favourites from inside the app
  - Recently used artifacts are now shown in a widget
  - …more can be found in the above-mentioned tool
- **Adaptive Templates**
  - Additional text elements can now be translated
  - Possibility to import CSVs with or without an id field
  - Adaptive Designer has been updated with new icons
  - Data Import Wizard for easier import of JSON, CSV and Planet9 files
  - Allow using of % wildcard character in the live search
  - New guided mode when creating a new application or events
  - …more can be found in the above-mentioned tool
- **App Designer**
  - Code snippets have been refactored and updated with more instructive comments
  - Show or hide deprecated attributes
  - Add objects to favourites in the library tree
  - … more can be found in the above-mentioned tool
- **Script Editor**
  - Calling API from server script is now configurable via “axiosConfig”
  - Now able to compare 2 server scripts
  - Naia is now available as a side panel
  - … more can be found in the above-mentioned tool

3.1.2. Basic usability and accessibility

3.1.3. Re-verification of bug fixes

The list of GitHub issues with fixes can be found in the tool mentioned above.

3.2. Limitations

- Performance Testing
- Stress Testing
- Database Testing
- GitHub changes and commits not testable from tester's perspective and only developers can verify such as server configuration files

**Not on regular basis but will be performed if required*

4. TESTING STRATEGY

A separate document is created for this section which you can find it here:

Neptune DXP Open Edition - Test Strategy.docx

5. TEST CRITERIA

5.1. Suspension Criteria

If there is at least one major functionality that is not working, for example, unable to create a mobile build then suspend the release of the build until the development team fixes it.

5.2. Exit Criteria

All test cases are performed with a mandatory of at least 80% PASS rate

6. RESOURCE PLANNING

6.1. System Resource

Resources	Description
Server	Installation of Neptune DXP Open Edition on a cloud
Automated test scripts	These scripts are developed using Playwright. Needed to perform the regression testing.
GitHub access	GitHub is used for bug tracking. If necessary, an access must be provided to all members of the QA Team

6.2. Human Resource

Resources	Tasks
Testing Guild **Leader**: Jens-Uwe Groß **Members:** • Rommel Lamanilao • Leonie Saremba • Penelope Labram • André Carrilho	• Manages the entire testing process • Manages and tracks documentation related to new functionalities • Creates and maintains the test plan and test strategy • Prepares test cases, provides more detailed information on how test and verify
Software Tester • Official names involve in this LTS testing are shown in a separate document	• Executes the test cases • Reports bugs/issues found to GitHub • Verifies bug fixes • Creates and maintains test cases
Automation Tester	• Develops and maintains automated test scripts • Executes automated test scripts • Manages end-to-end tests on CI/CD pipeline

6.2.1. Test Responsible

Here we define the list of testers who will be assigned to perform the testing for Neptune DXP — Open Edition ***(Testers have already been identified but still subject for proper assigning on which edition)***

- TBD
- TBD
- TBD

7. TEST CASE DESIGN

Test cases should be defined to determine if different features within a system are performing as expected and to confirm that the system satisfies all related standards, guidelines, and customer requirements. This process can also help reveal errors or defects within the system.

Test Management tool – all test cases are defined and created in this internal tool.

8. TEST ENVIRONMENT

The following should be setup to ensure software testing process success.

8.1. Test Server Setup

Install the Neptune DXP Open Edition the cloud. Installation type should be upgradable from latest master. We have already defined our test system which can be accessed through this -- https://p9latest.neptune-software.cloud/

8.2. Device

Tests will be performed on the following devices:

- Desktop
- Tablet
- Phone

8.3. Browser

The following browsers should be installed:

- Google Chrome
- Mozilla Firefox
- MS Edge
- Safari

9. SCHEDULE AND ESTIMATION

9.1. Test Estimation

Testing	Estimate Effort (%)
Smoke Test	10
Integration Test	40
System Test	50
TOTAL	100

9.2. Test Schedule

	Planned Date	Actual Date
Test started date	11-September-2024	
Test completed date	31-October-2024	

10. RISKS AND MITIGATIONS

Risks	Mitigations
Lack of test resource availability	Facilitate hiring or loan people from another department
Shortage of time to cover all the scope	• Focus on the new features and major changes • Focus on critical areas
Defects and bugs are found at a late stage and consumes time to resolve	• Defect management plan is in place to ensure prompt communication and fixing of issues

11. TOOLS

Tools	Usage
GitHub	For reporting of issues found during the testing and for requesting for a new feature – only for testers who have access to Planet 9 GitHub issue repository
Playwright	For creating automated test scripts
Test Management	This tool is intended for testers who do not have access to Planet 9 GitHub issue repository

12. BUG REPORTING

All encountered issues and feature requests should be reported to GitHub.

12.1. Bug Priority

Bug priority refers to how urgently a bug needs to be fixed. When creating a bug report, one must provide a priority on it. Although GitHub does not have a field for this priority, labels have defined instead.

Levels of bug priority defined as labels in GitHub:

- **Low:** Bug can be fixed at a later date. Other more serious bugs take priority
- **Medium:** Bug can be fixed in the normal course of development and testing
- **High:** Bug must be resolved at the earliest as it affects the system and renders it unusable until it is resolved

12.2. Bug Category

Like in bug priority, GitHub does not have category field as well so we will use labels instead. Category refers to the module or component where the bug is found. This is not a mandatory field but it is nice to have for easy filtering of the issue list.

These are the commonly used labels for categories:

- Use this label for general category particularly for smaller components/applications
    - Cockpit:

- For bigger components/applications:
    - Launchpad
    - App Designer
    - App Editor
    - Script Editor
    - Adaptive Designer
    - Deployment
    - Custom Component

When reporting an issue in GitHub, one must use **LTS-Testing** label. This would help us track list of issues reported during this LTS testing.