Testing @ the Austrian Post - A Practical Guide

Contents

1. The Purpose of this Document

This document is based on the Testing Handbook. However, it focuses on the practical side of testing, which is to say, the daily business of a test engineer at the Austrian Post.

It aims to offer an overview of practical testing and links to more comprehensive and in-depth information on the web (the KIT/QA Community Wiki). It is intended to serve as a starting point in this regard.

1.1 Version History

Version	Release Date	Amendment/s	Author(s)
1.1	2024.06.11	Added Ranorex and Tosca Minor updates and text fixes	Erik Fischelschweiger, Peter Fußl
1.0	2024.04.12	Initial version	Erik Fischelschweiger, Peter Fußl

2. Testing Guide

We use **Azure DevOps as the primary tool** for implementing the Agile development process outlined in the Testing Handbook. This chapter describes the test and QA relevant use cases as part of the Agile development process at the Austrian Post.

2.1 Testing in Iterative Development

In general, the Austrian Post uses an **iterative development** framework. Most teams use Scrum, but some also use Kanban or Scrumban.

Test Planning

Work Item Type: Feature, User Story

Work Item Status: New / Specified / Ready for Estimation

Outcome: The test approach as to how to test the work item has been defined

Test planning should be started **as early as possible**. In general, if Features or User Stories are created as part of a refinement, test engineers must consider testing the Feature or User Story. Therefore, it is important to challenge the requirements regarding testability, completeness, dependencies and

so on. Test engineers must keep in mind the testing complexity, the creation of test data, and test automation for the estimation.

Test Specification

Work Item Type: Feature, User Story

Work Item Status: Estimated / Active

Outcome: Test cases have been created in a test plan and linked to the corresponding requirement

The specification of test cases should be started at the latest when the items are in the current iteration. Therefore, test cases are created using a reasonable **step-by-step description** (i.e. test steps). A description of how test data will be generated and what else is needed to test the requirement must also be given. Besides functional tests, non-functional tests should also be in place. Furthermore, it should be decided whether automation is planned for a test case. If so, the automation status "Planned" is set and test automation can be started. Test automation is finished once the automated tests have been executed (see below). As soon as the automated test case has been linked to the 'Test Case' work item, the Automation status changes to 'Automated'.

Tests at the Feature level including system integration (or end2end-) tests are defined together with other systems/applications/teams if required.

Test Execution

Work Item Type: Feature, User Story

Work Item Status: Ready for Test

Outcome: Test cases have been executed, results published, and automated tests are running in a pipeline

If the work item's status is 'Ready for Test', a check must first be made as to whether all the criteria have been fulfilled and the test can be started. Test Cases are **executed manually** and, if planned, it is ensured that the automated test cases run in a pipeline regularly. Test execution can only be considered done when all automated tests pass regularly or - if failing – bugs were reported for failing test cases.

System integration- and end2end tests are executed within the team or together with other teams.

Analysis and Reporting

The **analysis of the outcome** of test automation and executing **regression tests** are essential as part of testing new User Stories or Features. **Communicating results** is an important part of a test engineer's work.

Daily Business and Maintenance

A test engineer's daily business requires that he or she always be aware of the status of all the team's products and the outcome of the nightly automated regression tests. Manual test cases and automated regression tests must **always** be **maintained and improved**. Furthermore, test engineers should check the state of the test environment, check logs, and do further explorative tests.

2.2 Testing Using Azure DevOps

The Austrian Post uses Azure DevOps as its iterative development framework. This implies that all work items – like Features, User Stories and Test Cases – are maintained within Azure DevOps. Test automation is hosted in the repositories in Azure DevOps and is triggered via 'Pipelines'. Test automation results are also stored in Azure DevOps.

Test Planning

A Test Plan in Azure DevOps can represent a project, a product, a P.R.I.M.E. train or a team. Test Suites can be used to structure it depending on the categorization of the Test Plan.

There are **three different types of Test Suites**:

- Static Test Suite: mostly used for structuring in teams, releases, regression-sets, functionalities, etc.
- Requirement-based Test Suite: represents User Stories or Features and includes Test Cases which are linked as 'Tested by'.
- Query based Test Suite: adds Test Cases for the result of a query to the Test Suite.

Test Specification

The main work as part of test specification is done within Test Cases. A Test Case can be created via a link in the User Story, the Feature or directly in a Test Suite. It is essential to **describe the steps** of a Test Case in a way that is **understandable** enough for another person to be able to execute the Test Case. Furthermore, preparing and generating test data is important and should already be done prior to test execution.

Using **tags** like 'Automatable' or 'Not test relevant' for User Stories and the automation status 'Planned' in Test Cases facilitates the monitoring of the status of Test Cases.

Further information: https://dev.azure.com/postat/KIT/_wiki/wikis/KIT.wiki%2F8585/Tagging-convention-for-testing#

Test Execution

Test Cases are not executable. Add a Test Case to a Test Suite to generate Test Point(s). After the Test Cases have been executed, a test result/outcome must be set in the "Execute" tab. The **documentation of test outcomes is vital**, otherwise it is not possible for everyone to see what has been tested and the test execution's result. Furthermore, logs and AppInsights must be checked if there are any (new) exceptions or problems.

Further information: https://learn.microsoft.com/en-us/azure/devops/test/navigate-test-plans?view=azure-devops#execute-tests

Test automation must be created for Test Cases where it has been planned. Therefore, Visual Studio is used, and **automated tests must be linked to the (manual) Test Cases** in Azure DevOps so that the automation status of the Test Cases is set to 'Automated'. Test automation runs nightly via Azure DevOps Pipelines. Consider creating a ReadMe file in your repository containing information for others on how to run the project locally.

Test Reporting

There are several possibilities for test reporting in Azure DevOps depending on their automation status.

Manual Test Cases:

- Check the results in Test Plans.
- Check the results in Dashboards via queries/widgets.

Automated test cases:

- Check the results in Pipelines.
    - Advice: subscribe to the outcome of pipeline runs to get notified via email in Azure DevOps User Settings/Notifications
- Check the results in Test Plans.
- Check the results in Dashboards via queries/widgets.

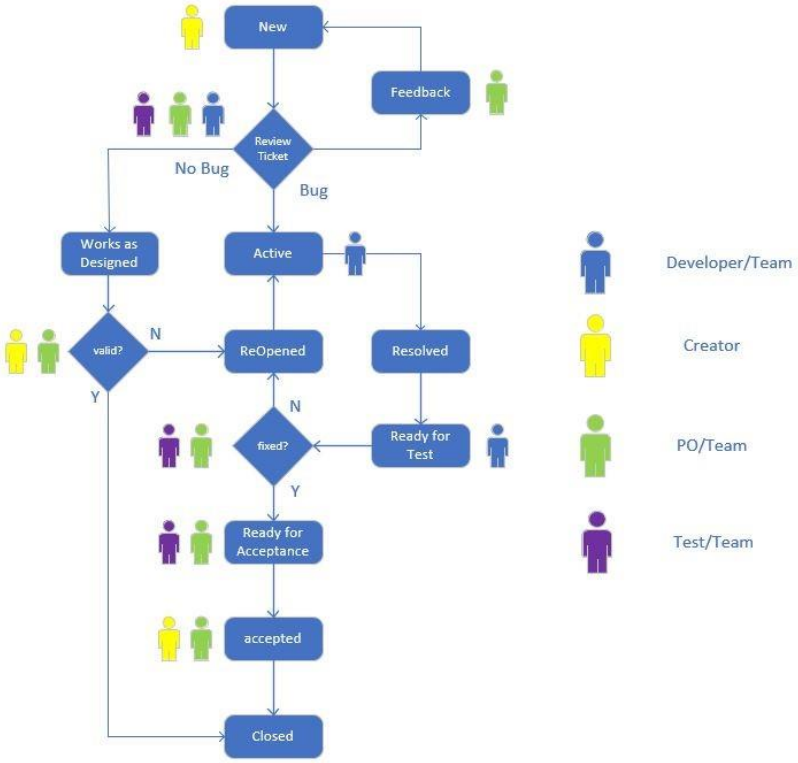
2.3 Bug Management

The work item type 'Bug' is used in Azure DevOps for deviations. It is essential to **document deviations in a precise and reproducible way** so that developers or product owners can decide how to proceed.

Further information on how to handle Bugs (which fields need to be filled by whom, severity, priority etc.) can be found in the QA Community Wiki: https://dev.azure.com/postat/KIT/_wiki/wikis/KIT.wiki/8556/Bug-Guide

A Bug's Lifecycle

The management of the lifecycle of a Bug is based on the following workflow:



In addition to the statuses displayed in the diagram above, there is the 'Tested' status that can be used but is not mandatory. Basically, it is used to visualize that the bug was retested and fixed.

If bugs have been remedied and are ready for testing, the following tests must be performed:

- A **bug re-test** to check whether the defect has in fact been remedied.
- A regression test to ensure that no new bugs/deviations have arisen because of the fix.

3. Testing Infrastructure

This basic rule applies to all projects, products, services, and applications:

At a minimum, **DEV-, TEST-, ABN (= acceptance)- and PROD-stages** are necessary and need to be set up for all new applications and services. These stages are usually set up/requested by operations during projects run by the dev-teams.

Note: old or legacy systems like VDE do not always stick to this pattern, unfortunately.

Testing (manual and automated) is usually done at the TEST stage, whereas the DEV stage tends to be too unstable (due to a lot more deployments/releases by programmers) for proper testing.

The ABN stage is the most stable non-PROD environment for an application. Acceptance tests are usually conducted at this stage. As the ABN stage usually contains the status of the application being deployed to PROD, it is recommended that the test automation also be run on ABN (in addition to the TEST stage).

Test automation pipelines are usually run in the **Test Agent pool**. Some test automation projects run in the shared build pool. However, this is not advised. More information about the test agents can be found here in the QA Community wiki: https://dev.azure.com/postat/KIT/_wiki/wikis/KIT.wiki/45154/Test-Agents-Overview

Ideally, test automation pipelines are set up in the KIT project in Azure DevOps and are scheduled to run at least nightly. An introduction on how to set up a pipeline for test automation can be found here: https://dev.azure.com/postat/KIT/_wiki/wikis/KIT.wiki/24791/Set-up-pipeline-for-test-automation-project-draft-

Depending on the application being tested, different requirements might need to be installed or placed on the test agents. The most common/most frequently used requirements for test agents can be found here: https://dev.azure.com/postat/KIT/_wiki/wikis/KIT.wiki/31554/Test-Agent-Requirements

4. Test Data Generators

There are several self-made test data generators being used in different teams and areas at the Austrian Post. This chapter describes what they were made for and how they are being used.

4.1 ELFRIEDE

Scope of application: The creation of users and parcels in the ABN environment

Used in teams/areas: Post Web, Post App, ...

ELFRIEDE is a website used to **create users** (i.e., online Post accounts) **and parcels** (with and without a link to users), to search for existing users, and to change the status of shipment **in the ABN environment**. Private users can be created with or without identification.

ELFRIEDE also has an API interface that can be used for test automation.

Further information: https://dev.azure.com/postat/KIT/_wiki/wikis/KIT.wiki/4963/Test-Data-Generators?anchor=elfriede

4.2 VDE Test Center

Scope of application: The generation and manipulation of parcel events

Used in teams/areas: LOG Train (VDE)

In the VDE Test Center, all the VDE read and write **web service methods** are offered for the purpose of querying or generating events and thus importing them for **parcels**. It is possible to import shipments from PROD and to export shipments into an EVL file to reuse it. Access to a specific AD group is needed to use the VDE Test Center.

Further information: https://dev.azure.com/postat/KIT/_wiki/wikis/KIT.wiki/4963/Test-Data-Generators?anchor=vde-test-center

A list of shipments exported to EVL to reuse for test scenarios can be found here: https://dev.azure.com/postat/KIT/_wiki/wikis/KIT.wiki/39473/Collection-of-Shipments-for-Tests

4.3 TEDDI

Scope of application: The import of shipment data from production

Used in teams/areas: DISTRI Train

TEDDI is a tool integrated in ELFRIEDE that can be used to **import shipment data** from production to a testing environment. The data is anonymized in the process.

Further information: https://dev.azure.com/postat/KIT/_wiki/wikis/KIT.wiki/16737/TEDDI

4.4 Test Data Generator (TDG)

Scope of application: The generation of barcodes for scans of the IRIS handhelds

Used in teams/areas: DISTRI Train

The Test Data Generator (TDG) can be used to **generate various barcodes** for the scans of the IRIS handhelds such as letters, packages, pick-up orders etc.

Further information: https://dev.azure.com/postat/KIT/_wiki/wikis/KIT.wiki/4963/Test-Data-Generators?anchor=tdg---test-data-generator

4.5 UTE

Scope of application: The automatic generation of test shipments in a pipeline

Used in teams/areas: Post App, Post Web

UTE utilizes the **VDE Post.TAF NuGet** and its keywords to **create shipments** for testing purposes. It can be set up in pipelines to create shipments regularly for the purpose of solving the expiration problem.

Further information: https://dev.azure.com/postat/KIT/_wiki/wikis/KIT.wiki/41859/UTE-Utility-Test-data-Extension

4.6 KLAUS

Scope of application: The generation and manipulation of Post online test accounts

Used in teams/areas: KIAM, Post Web, ELFRIEDE

KLAUS can be used to **generate users** (i.e. Post Online accounts) in KIAM, **manipulate user data**, complete double opt-ins, etc.

Further information: https://dev.azure.com/postat/KIT/_wiki/wikis/KIT.wiki/4963/Test-Data-Generators?anchor=klaus

4.7 Additional Post-Internal Tools and Apps

There are several Post internal applications that can be used for testing purposes (to create data, to check data, etc.), ranging from team-internal to back-office apps that are also used in production.

A list of tools and more information can be found here: https://dev.azure.com/postat/KIT/_wiki/wikis/KIT.wiki/44644/Additional-Post-Internal-Tools-and-Apps

5. Manual Testing Tools

Several tools are used for manual testing purposes at the Austrian Post. The following list briefly describes the most common tools, offers links to further information/official documentation, and where (team/area) the tools are being used within the Austrian Post.

5.1 Postman

Scope of application: Testing APIs

Used in teams/areas: Shared Services, KIAM, ...

Postman is a tool for manually testing APIs (REST, Soap, GraphQL). It is possible to build up collections of such API requests (and even testcases) for sharing within teams and applications. Postman is not used as an automation tool at the Austrian Post.

Documentation: https://learning.postman.com/docs/introduction/overview/

5.2 SQL Server Management Studio

Scope of application: Testing database contents

Used in teams/areas: Anubis, Allround Squad, ...

SQL Server Management Studio (SSMS) is used for manually testing the contents of SQL databases and/or to prepare and manipulate test data.

Documentation: https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16

5.3 Kafkaesque

Scope of application: Testing Kafka events

Used in teams/areas: Tanuki, ESP, ...

Kafkaesque is a desktop application which is useful for inspecting actual published events and schemas. It can be used without command line tools to send messages into topics/clusters.

Further information: https://dev.azure.com/postat/KIT/_wiki/wikis/KIT%20HoneyPot/34528/Kafkaesque

5.4 ServiceBus Explorer

Scope of application: Testing Service Bus topics and messages

Used in teams/areas: Shared Services

The Service Bus Explorer allows users to connect to a Service Bus namespace and test sending, receiving, and peeking messages. There are two variants: a 3rd party application and directly in the Azure Portal (in its preview-phase at the time of writing).

Further information (3rd party application): https://dev.azure.com/postat/KIT/_wiki/wikis/KIT.wiki/9960/Service-Bus-Explorer

Documentation (in Azure Portal): https://learn.microsoft.com/de-de/azure/service-bus-messaging/explorer

6. Test Automation Tools

There are several test automation tools being used at the Austrian Post. This chapter provides an overview of them and links to further information about the tools, how to set them up, which teams are using them, etc.

6.1 Post.TAF

Scope of usage: Test automation

Used in teams/areas: DLP, Shared Services, Post App, ...

Multilayer test automation can be performed with Post.TAF. It includes clients for individual layers (GUI, web service, database). However, the Selenium (Web) Client is basically the only one being used actively. For other layers (esp. APIs, Kafka, etc.), the framework is used but not the provided (i.e. 'built-in') clients.

Post.TAF and the individual clients can be downloaded from the Austrian Post's feed via packages. At the same time, each individual test solution and, as a result, all the keywords, data providers etc. contained therein, can be made available as a NuGet package, and re-used by all other teams/test engineers. Here is a list of NuGet packages for the different applications provided: https://dev.azure.com/postat/KIT/_wiki/wikis/KIT.wiki/44509/TAF-NuGets-Overview

How to set up a new Post.TAF project is described in detail here: https://dev.azure.com/postat/KIT/_wiki/wikis/KIT.wiki/6674/How-to-set-up-a-new-TAF-project-with-example

Selenium – Web Driver Client

Scope of usage: UI web application test automation

Used in teams/areas: LOG train, ...

The Selenium open-source framework is used for the automation of web applications. In the process, the necessary Selenium WebDriver components are integrated into the development environment using NuGet packages.

Documentation: https://www.selenium.dev/documentation/

6.2 Playwright

Scope of usage: UI web application test automation

Used in teams/areas: Post Web, ELLA, DIAL, BELLA

Playwright is a framework for web (GUI) test automation. It allows testing in Chromium, Firefox and WebKit with a single API.

Documentation: https://playwright.dev/docs/intro

6.3 Ranorex

Scope of usage: Desktop Client test automation

Used in teams/areas: OPAL

Ranorex is a framework for functional UI test automation. It allows testing desktop, web and mobile applications.

Documentation: https://support.ranorex.com/userguide/ranorex-studio-fundamentals/

6.4 Tosca

Scope of usage: SAP test automation

Used in teams/areas: SAP

Tosca is a framework for functional and non-functional test automation. It allows testing desktop, web and mobile applications, APIs, databases etc.

Documentation: https://support-hub.tricentis.com/open?id=tosca_manuals

7. Performance & Load Testing

The need for performance and load testing depends on the application. As a rule of thumb, load and performance tests are done before initial go-lives if required (i.e., there are high loads expected and/or there are non-functional requirements defined for certain max. load/response times and the like). These tests can of course also be set up in pipelines and run regularly if needed.

7.2 Tools

With respect to load and performance tests, there are basically three tools being used at the Austrian Post: JMeter, Azure Load Testing and K6.

JMeter

Scope of application: Load and performance tests

Used in teams/areas: Allround Squad, Tanuki, ...

JMeter is used to perform load tests on client-server applications. JMeter can be used in many ways. These facilitate, among other things, the application and execution of existing TA tests via command line. JMeter is usually used on local machines; Azure Load Testing is the tool to use for generating load.

Documentation: https://jmeter.apache.org/usermanual/index.html

Azure Load Testing

Scope of application: Load and performance tests

Used in teams/areas: Allround Squad, Tanuki

Azure Load Testing can be used for simple load testing use cases. However, it is possible to import JMeter scripts for more complex scenarios. The tool is also used to generate load (exceeding the load generated running on local machines).

Further information: https://dev.azure.com/postat/KIT/_wiki/wikis/KIT.wiki/37656/Azure-Load-Testing

K6

Scope of application: Load and performance tests

Used in teams/areas: Post Web, KIAM

K6 is an open-source tool for load and performance tests and is especially useful for web application developers as it uses JS.

Documentation: https://k6.io/docs/

8. Accessibility Testing Tools

Accessibility testing for public web sites and apps is an important part of the testing process at the Austrian Post. Accessibility testing allows you to check if your web site or application can be used by people with disabilities such as visual or hearing impairments.

8.1 WAVE Evaluation Tool

Scope of application: The evaluation of accessibility in browsers

Used in teams/areas: OBS

The WAVE Evaluation Tool can be used to evaluate web accessibility within browsers. It is a plugin for the local Chrome browser. It can also be used for websites or stages that can only be reached internally.

Documentation: https://wave.webaim.org/

8.2 Lighthouse

Scope of application: The evaluation of accessibility (but also SEO, performance etc.) in browsers

Used in teams/areas: Post Web

Lighthouse is a tool for improving the quality of web pages. It can audit performance, accessibility, SEO, and more. It can be used via Chrome or node installation.

Documentation: https://developer.chrome.com/docs/lighthouse/overview

8.3 Playwright

Scope of application: Testing the accessibility of a website

Used in teams/areas: Post Web

Playwright is an automation framework which uses the Axe Accessibility Testing Tools package for running accessibility tests on websites. The result can be exported and reviewed. As stated in the documentation, automated accessibility testing can be helpful, but should not serve as a substitute for manual accessibility testing.

Documentation: https://playwright.dev/docs/accessibility-testing

8.4 Screen readers

Scope of application: Test text to speech

Used in teams/areas: OBS

Screen readers are used to render text and image content as speech or even as braille output. There are browser plugins or even built-in screen readers (e.g. for Windows) that can be used for testing purposes.

Documentation: https://support.microsoft.com/en-us/windows/complete-guide-to-narrator-e4397a0d-ef4f-b386-d8ae-c172f109bdb1