DOCUMENT

Executive Summary Report

TAEF – Test Automation Exchange Framework

Prepared by	**TAEF team**
Reference	**77-057_TAEF_ESR_ExecutiveSummaryReport**
Issue	**1**
Revision	**0**
Date of Issue	**2022-12-16**
Status	**Final**
Document Type	**TN**

European Space Agency
Agence spatiale européenne

Distribution

APPROVAL

Title Executive Summary Report	
Issue 1	**Revision 0**
Author TAEF team	**Date 2022-12-16**
Approved by	**Date**

CHANGE LOG

Reason for change	Issue	Revision	Date
Initial draft	0	1	2022-12-01
Version for review	0	2	2022-12-06
Version for AR	1	0	2022-12-16

CHANGE RECORD

Issue	Revision		
Reason for change	**Date**	**Pages**	**Paragraph(s)**

Table of contents:

1 INTRODUCTION

1.1 Purpose

This document presents the final report containing a summary of the tasks performed during the course of the "TAEF – Test Automation Exchange Framework" activity and the produced results.

1.2 Scope

This document constitutes deliverable FR, linked to the outcome of "WP 800– Management". It covers the entire scope of the activity, providing a complete description of all work performed, giving an overview of the context of the activity and the achieved results.

1.3 Document Overview

Section 1 - Introduction (this section) provides the purpose, scope, and this document's overview.
Section 2 - Applicable and Reference Documents provides the list of reference documents.
Section 3 - Terms, Definitions and Abbreviated Terms provides a list of acronyms and terms used throughout this document.
Section 4 - Executive Summary presents the summary and the results of the activity.

2 APPLICABLE AND REFERENCE DOCUMENTS

2.1 Applicable Documents

Ref.	Document Title	Issue and Revision, Date
[AD-1]	Technical Proposal – Test Automation Exchange Framework (TAEF)	1.0, 03.03.2021
[AD-2]	Statement of Work ESA Express Procurement [Plus] - [EXPRO] [EXPRO+] – Test Automation Exchange Framework, AO/1-10454/20/D/MRP GT1O-311GD	1.0, 04.12.2020

2.2 Reference Documents

Ref.	Document Title	Issue and Revision, Date
[RD-1]	EUDART-SDD-2.0	2.0
[RD-2]	EUDART-SUM-2.0	2.0
[RD-3]	EUDART-CG-2.0	2.0
[RD-4]	LANGOSTA-SSDD-0001	1.8, 2020.07.06
[RD-5]	LANGOSTA-SUM-0001	1.2, 2020.08.03
[RD-6]	LANGOSTA-TN-0003-MiconysCC-Tests	2.0, 2020,09.30
[RD-7]	TAEF – Final Report	1.0, 2022.12.16

3 TERMS, DEFINITIONS AND ABBREVIATED TERMS

3.1 Acronyms

Acronyms	Description
ART	Automated Regression Testing
CLI	Command Line Interface
E2E	End-to-End
GSRF	Ground Segment Reference Facility
EUD	Egos User Desktop
ESA	European Space Agency
ESOC	European Space Operations Center
MMIT	Man-Machine Interface Testing
RCP	Rich Client Platform
SoW	Statement of Work
SUT	System Under Test
SWT	Software Widget Toolkit
TAEF	Test Automation Exchange Framework
TTI	Test Tool Interface
UML	Unified Modelling Language

3.2 Definition of Terms

Terms	Description
EUDART	ESOC software based on EUD that controls the execution of GSSW functional tests
EGS-CC	European initiative to develop a common infrastructure to support space systems monitoring and control in pre- and post-launch phases for all mission types.
TAMDEM	Demonstrator for Test Automation using Graphical User Interfaces (study)
laNGoSTA	New Ground Segment Test Automation (study)

4 EXECUTIVE SUMMARY

4.1 Overview

The "TAEF – Test Automation Exchange Framework" GSTP activity started in July 2021 and was concluded, after 18 months, in December 2022.

The overall work logic of the activity was broken down into the following tasks:

- WP100 – EUDART Extension - Creation of TCs
- WP 200 – EUDART Extension - OTX compatibility
- WP 300 - EGS-CC GUI test library development
- WP 400 - EGOS-CC components test library development
- WP 500 – Test orchestration improvements
- WP 600 – Space model concept integration analysis
- WP 700 – TMS integration improvement

The produced deliverables were the following:

- Update to EUDART SUM
- Update to EUDART CG
- Update to EUDART SDD
- Technical Note on OTX parsers
- SW1 – EUDART (update and new functionality)
- SW2 – OTX parsers

4.2 Background

The driver behind the TAEF activity was to propose ways to improve the test automation framework and workflow used at ESOC with the new core infrastructure (EGS-CC) in mind. Test automation is critical for the successful maintenance and evolution of large software projects. The automation of the tests can take place at different levels, ranging from unit testing to system testing. While low level test automation can be achieved with programming techniques (or equivalent), the automation of system level testing requires a more elaborate approach. System level tests are typically written by the people who will operate the software in natural language (which is often imprecise and quite loose), but, at the same time, the automation system must understand them.

4.3 Objectives

The main objective of this activity was to update the existing test automation tools and framework, make it fit to validate the next generation of mission operations software (EGS-CC and web-based) and provide additional enhancements to facilitate collaborations with other test automation frameworks using the OTX format.

4.4 Test Actions Generation

The goal of this epic was to enhance EUDART to allow it to create and modify test actions, test cases and test schedules in the XML for EUDART format. These types of actions are currently maintained by TEMPPO Designer, which is a proprietary tool and therefore some of its functionality needs to be reimplemented.

A mechanism to create and maintain test actions was designed as an extension to EUDART in the form of a new view called “Library View” that allows to open next to a test schedule in the “Runner View”, one or more test libraries to support with the creation of test cases.

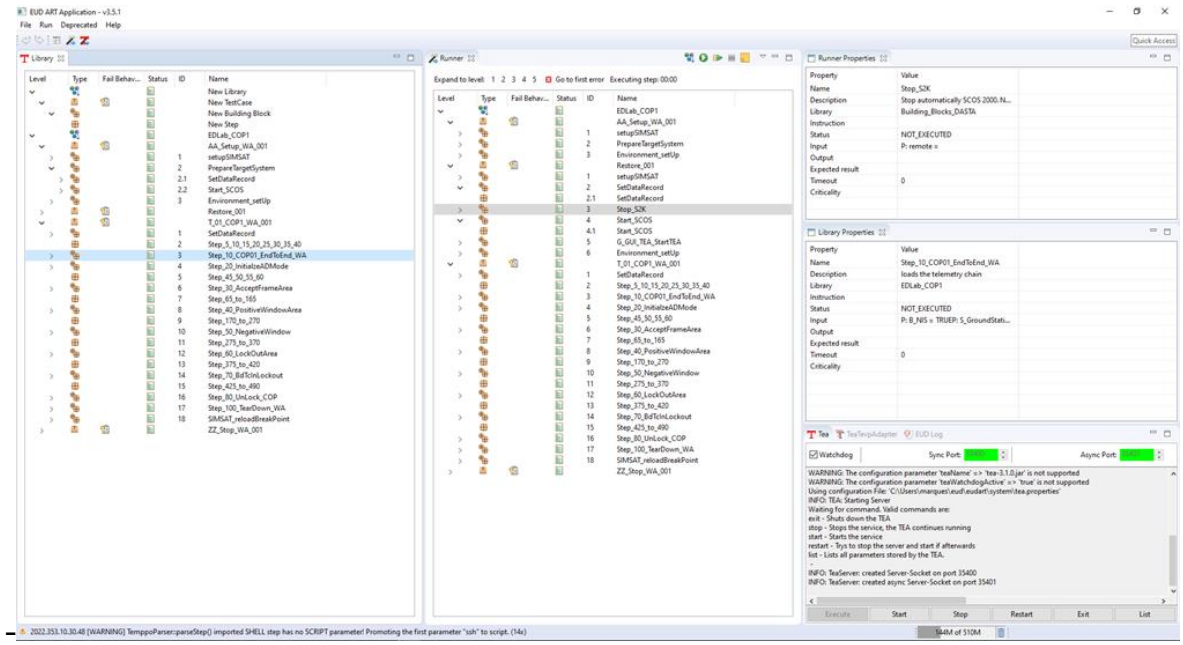

Figure 1: EUDART library view

The “test libraries” that can be imported into EUDART are not same format as the original test libraries in TEMPPO. For a test library to be imported into EUDART it must be exported in TEMPPO Designer as a “test schedule”. This decision was taken because of the underlying data model of EUDART. This makes, from a EUDART perspective, everything the same format (test libraries and test schedules) and allows to directly import a test schedule as a

test library in to EUDART for reuse and to support with the creations of test cases. This approach makes creating new test cases from existing test cases possible (instead of always having to use test libraries) and simplifies the workflow of the test engineer. Instead of importing test libraries it is also possible to use a default library that contains empty test step, test case and building block elements that can be used by the user to create new test steps, test cases or building blocks.

The new library view allows the user to drag-and-drop elements to the runner view to create or modify test cases. Alternatively, it is also possible to use the right-click context menu to add new building blocks or test steps to the runner view.

4.5 Test Actions Exchange using OTX

An investigation was performed to analyse how to exchange test actions between different testing frameworks. The OTX format for being used in other tools like ATENA and for being a standardised format was used for this purpose. Two parsers were generated (derived from the ATENA tool) to convert a project saved in the IDATG format to the OTX format and from the OTX format to the EUDART XML format

- atena-idatg-parser (conversion from IDATG to OTX)
- atena-eudart-parser (Conversion from OTX to EUDART XML)

Both parsers can be used from CLI and were developed by establishing a mapping between the OTX format and the IDATG and EUDART XML formats.
Both parsers are prototypes and are a partial mapping between the formats. (Not all inputs have a 1-to-1 match in the converted format)
These prototype parsers can be used as input to in future work activities implement the conversion/parsing directly into EUDART if this is deemed beneficial. At this moment it is possible to use the parsers as standalone tools.

4.6 Development of test libraries

A set of new test libraries were developed in the scope of this activity defining both generic and application specific test actions. Two types of libraries were generated

- Test libraries for EGOS-CC (with example tests for BEPI and JUICE, focused on FBO, TFI and ASA components)
- Test libraries for the web-based version of EGS-CC

These libraries translated into the following repositories in Gitlab

- LibItemsWebUI (https://gitlab.esa.int/eudart/LibtemsWEBUI)
- LibItemsMiconysCC (https://gitlab.esa.int/eudart/LibItemsMiconysCC)
- LibItemsGen (https://gitlab.esa.int/eudart/LibItemsGEN)

And are supplemented by the following repositories with the respective test items:

- TestItemsJUICE (https://gitlab.esa.int/eudart/testitemsjuice)
- TestItemsBepi (https://gitlab.esa.int/eudart/testitemsbepi)
- TestExecutionEGOSCC (https://gitlab.esa.int/eudart/TestExecutionEGOSCC)

The generic testing library was created to support the testing of web-based UIs using the ART-MMIT framework. The rationale behind extending the ART-MMIT framework was to reuse the test automation infrastructure and to keep the workflow identical to the creation of tests for EUD based application like S2K. The extension to the ART-MMIT framework was implemented as a new TTI called SeleniumTTI. These new libraries encapsulate basic actions understood by ART-MMIT and translates them into action understood by Selenium for the execution.

4.7 Test Orchestration and deployment

This task aimed originally at improving and updating the test orchestration and deployment of the test automation framework related components according to the new artefacts generated and updates to existing tools and artefacts resulting from this study.
This was performed together with a set of updates to several components of the test automation framework listed below

- Creation of CI/CD pipeline for EUDART in Gitlab for build and deployment
- Update of the EUDART dependencies to EUD4 from EUD3
- Update of the EUDART Java version from 8 to 11
- Update to SLES15 baseline from SLES12 baseline
- Definition of Eclipse-RCP target platform for EUDART based on Nexus and the systems accepted repository

- Alignment of headless maven build of EUDART with the development environment RCP build through target platform
- Update to EUDART build to generate linux and windows executables for EUDART
- Update to components of the ART-MMIT framework to allow the use of generic TTIs using reflections
- Bug fixes and improvements/updates in products contained in the MMIT repository (GTD, TEA, SWTBotTTI)

4.8 Scope changes

The tasks “Compatibility with TMS” and “Analysis of Integration with the Space System Model” were dropped, and the allocated efforts were used in the creation of new test libraries as this was considered more valuable to the outcome of the study. In the case of the Space System Model Integration, these tasks was already being investigated in the MoBaTe - “Enabling Model-Based Testing and Automated Test Case Generation for Ground Segment Data System” study.

4.9 Results and way forward

A new version of EUDART was produced that is based on a more recent version of EUD and Java 11 and that includes a new library view to support the user with the creation of test cases and test schedules without having to necessarily use TEMPPO Designer every time. This approach is a proof of concept and is suitable for simple test cases and test schedules created using other test cases and test schedules or using TEMPPO Designer libraries exported as test schedules. This prototype requires further validation to access the usability and the possible added value to the user, and also further development to incorporate additional functionality from TEMPPO.

Several new testing libraries and sample tests were created for both EGOS-CC and the web-based version of EGS-CC. These web libraries allow the user to use the ART-MMIT framework to test web applications. The libraries generated in the scope of this study need to be further developed to cover as many actions as possible from the actions made available by the SeleniumTTI as well as to create reusable fragments for specific applications.

The work performed during this study showed that EUDART is difficult to maintain and extend due to the lack of comments in the source code, big amount of deprecated code from deprecated functionality and the usage of the EUD dependencies which in some situations is not always helpful. It is clear that EUDART was originally designed to do more than it currently does but the code base was never refactored, making the code more difficult to read.