Runbook

for

For NASA Maestro Format Test Tool

**Version 1.0**

**April 26, 2020**

**Version 1.0**

**Prepared by:**

**NASA TEAM 1**

**University of Maryland Global Campus**

**19 April 2020**

Table of Contents

[1. Overview 4](#_Toc38716791)

[2. Build 4](#_Toc38716792)

[2.1 File Structure 4](#_Toc38716793)

[3. Deployment 4](#_Toc38716794)

[3.1 Standalone Project Deployment 4](#_Toc38716795)

[3.2 Docker Deployment 5](#_Toc38716796)

[4. Common Tasks 5](#_Toc38716797)

[5. Error Messages 5](#_Toc38716798)

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Date** | **Reason for Changes** | **Version** |
| 04/19/2020 | Initial Version | 1.0 |

# Overview

This NASA Maestro Format Test Tool project is a standalone application deployable locally or on a web server. The feature application is developed, deployed, and administered as a client/server application. The primary goal of the app is to ensure that .docx files created by Maestro are valid and then compare formatting between two Docx files. This run book targets the System Administrator and advanced users to detailed processes for maintaining and deployment of the application. The target user should be familiar with the following concepts: JavaScript, Git, GitHub, Docker, and Client-Side development.

# Build

The Maestro Format Test Tool Project employs Angular and TypeScript to create a simple local web base user interface; Visual Studio Code (VSCode) as a development environment; JavaScript, Electron, and Node.js as standard programming framework. Additional functionality is developed in JavaScript and Node.js hosted in a Docker Linux Container.

## File Structure

The application was developed using the client, server methodology. The server-side handles file manipulation and image comparison. The client-side serves a user interface to initiate application actions and retrieve manipulated or created files.

The application files are found in the source folder and are further split based on the above methodology with the server having a folder and the client having a folder.

# Deployment

The NASA Maestro Format Test Tool software could be deployed as a standalone project or as a Docker container application. The docker serve is designed to be standalone and provide additional functionality to calling application using HTTP requests as stated in the project’s API.

## Standalone Project Deployment

To launch the NASA Maestro Format Test Tool as a standalone application, the client’s administrator should copy the entire application’s project folder (with exception of Documentation and project-docs) to any platform that supports NodeJS (v12). The standalone deployment should be done following the steps stated below:

1. On the target machine, open a terminal/command-line console in the ${project root}/backend
2. Install project dependencies, execute the command
   1. npm install
   2. node downloads all dependent packages for the Angular server components
3. Start the server, run the following command
   1. npm start
   2. the server will start listening for requests on local port 3000
4. Utilizing user-provided server, navigate a browser window to ${server}/frontend/index.html
5. The application is ready for execution. The User Guide contains full instructions for execution.

## Docker Deployment

For docker deployment, the server and client should be a permeant set up on a server to provide access for users. The docker container can run on a local machine or remote machine. Likewise, the client UI can run locally or be run on a server.

# Common Tasks

Maestro Format Test Tool application is structured as a client/server web application. The application does not demand any administrative tasks to be performed while it is running.

# Error Messages

The current UI contains limited functionality and displays any error that is encountered on the results page as an error message. Sample error messages include:

* No image file returned. DOCX comparison failed.
* No file returned. DOCX validation did not complete. DOCX validation is inconclusive.
* No file returned. DOCX to image conversion failed.

The docker container will return HTTP error codes as the request fails. These include the following:

|  |  |  |
| --- | --- | --- |
| Request | Error Code | Response |
| Get Comparison Image | 404 | Session not found |
|  | 500 | Internal server error |
| Get Posted Image | 404 | File not found |
|  | 500 | Internal server error |
| Get Comparison Image as a Byte Stream | 404 | Session not found |
|  | 500 | Internal server error |
| Get API Main Page | 404 | Page not found |
| Validate a DOCX File | 400 | Bad request: less or more than 1 file uploaded |
|  | 500 | Internal server error |
| Compare Two DOCX Files | 400 | Either less than one or more than 2 files uploaded |
|  | 500 | Internal server error |
| Convert DOCX to Image | 400 | No files uploaded |
|  | 500 | Internal server error |