Official Documentation: MediaSphere EPUB Viewer

Version 1.0 | **Last Updated:** July 29, 2025

Abstract

The MediaSphere EPUB Viewer is a desktop application designed for a clean and immersive e-book reading experience. It is a key component of the open-source MediaSphere Suite, engineered with Electron.js to deliver a cross-platform solution for viewing EPUB files. The application leverages the powerful `epub.js` library for high-fidelity rendering and navigation. This document provides a comprehensive overview of the project's architecture, features, installation procedures, and component specifications.

1.0 INTRODUCTION

The MediaSphere EPUB Viewer offers users a focused and feature-rich platform for reading digital books in the EPUB format. As part of the broader MediaSphere Suite, it shares the same commitment to open-source development, performance, and user-centric design. The project is actively developed by a community of contributors, including members from GITAM (Deemed to be University).

The primary objective of the MediaSphere Suite is to create a single, unified application for all media formats. The EPUB Viewer represents the project's solution for digital literature.

2.0 KEY FEATURES

- **Fluid E-Book Rendering:** Displays EPUB documents with accurate formatting, images, and styles.
- **Interactive Navigation:** Allows users to move between chapters using a dynamic table of contents, or turn pages sequentially.
- **Customizable Reading Experience:** Features controls for adjusting font size to suit user preference.
- **Modern, Clean Interface:** A minimalist and dark-themed UI designed to minimize distractions and provide a comfortable reading environment.
- **Integrated File Management:** Users can open local EPUB files directly through a native file dialog.

3.0 INSTALLATION AND EXECUTION

3.1 Prerequisites

A working installation of Node.js is required to run the application.

3.2 Procedure

Execute the following commands in a terminal or command prompt:

Clone the source repository:
git clone https://github.com/AtheeqAhmedMJ/MediaSphereEPUBViewer.git

1.

Navigate to the project directory:
cd MediaSphereEPUBViewer

2.

Install dependencies:
npm install

3.

Execute the application:
npm start

4.

4.0 SYSTEM ARCHITECTURE

The application is built using the **Electron.js** framework, which enables the creation of desktop applications with web technologies. The architecture is bifurcated into two primary processes to ensure security and performance:

1. **Main Process (`main.js`):** The application's backend, running in a Node.js environment. It manages application windows and handles native operating system interactions.
2. **Renderer Process (`renderer.js`):** The application's frontend, responsible for rendering the user interface within a sandboxed Chromium window.

These processes communicate securely through a **Preload Script (`preload.js`)**, which selectively exposes backend functions to the frontend.

5.0 COMPONENT SPECIFICATION

This section details the function and design of each core file in the project.

5.1 Project Manifest (`package.json`)

This file serves as the project's configuration manifest, defining its metadata and dependencies.

- `"main": "main.js"`: Specifies the entry point for the Electron application.
- `"scripts": { "start": "electron ." }`: Defines the `npm start` command for easy execution.
- `"dependencies": { "epubjs": "^0.3.93" }`: Declares **Epub.js** as a critical dependency for rendering EPUB files.

5.2 Main Process (`main.js`)

This script controls the application's lifecycle and backend operations.

- **Function: `createWindow()`**
  - **Purpose:** To initialize and configure the main application window that the user interacts with.
  - **Action:** A `BrowserWindow` instance is created. The `webPreferences.preload` option is set to load `preload.js`, establishing a secure communication bridge to the renderer process.
  - **Result:** A native desktop window is created, ready to load the `index.html` user interface.
- **IPC Handler: `ipcMain.handle('dialog:openFile', ...)`**
  - **Purpose:** To provide a secure way for the UI to request access to the local file system.
  - **Action:** An Inter-Process Communication handler is established using `ipcMain.handle()`. When the UI requests to open a file, this function executes, showing a native file dialog filtered for EPUB files (`.epub`). It returns the path of the selected file to the renderer process.
  - **Result:** The user can securely select a local EPUB file without exposing the entire file system to the sandboxed UI.

5.3 Preload Script (`preload.js`)

This script acts as a secure bridge between the frontend and backend.

- **API: `contextBridge.exposeInMainWorld('electronAPI', ...)`**
  - **Purpose:** To securely expose specific backend functions to the renderer process.
  - **Action:** The `contextBridge` module attaches a custom `electronAPI` object to the UI's global `window` object. This object contains the `openFile()` function, which internally invokes the `dialog:openFile` IPC handler in the main process.
  - **Result:** The UI can call `window.electronAPI.openFile()` to trigger a file dialog, maintaining a strong security boundary between the two processes.

5.4 Renderer Process (`renderer.js`)

This script governs the application's user interface, managing the e-book rendering and user interactions.

- **Core Library: Epub.js**
    - **Purpose:** To parse, interpret, and render the contents of an EPUB file.
    - **Action:** The script imports the `epub` library. When a file path is received from the main process, an `ePub()` book object is created with that path.
    - **Result:** The complex structure of the EPUB file is parsed into a manageable book object.
- **Function: `book.renderTo("viewer", ...)`**
    - **Purpose:** To display the parsed e-book content on the screen.
    - **Action:** This core `epub.js` function takes the book object and renders it into the HTML element with the `id="viewer"`. It manages the layout, pagination, and flow of the e-book content automatically.
    - **Result:** The EPUB document is displayed to the user in a clean, readable, multi-page format.
- **Navigation and Customization**
    - **Action:** Event listeners are attached to UI buttons. Clicking "next" or "prev" calls `book.nextPage()` or `book.prevPage()` respectively. The font size slider interacts with the `book.themes` API to dynamically change the text size within the rendered view. The table of contents is dynamically generated by iterating through `book.navigation.toc`.
    - **Result:** The user has full control over their reading experience, with interactive navigation and customization options.

6.0 LICENSE