CONTENT

FUNCTIONAL DOCUMENTATION:

Additional Libraries:

Electron

Electron.js is a framework for building cross-platform desktop applications using web technologies - HTML, CSS, and JavaScript. You build your app like a website, and Electron wraps it in a native shell using Chromium (for UI) and Node.js (for backend logic). In short - great for fast, cross-platform development, but not ideal if needed lightweight performance or deep native integration.

Link - https://www.electronjs.org

SheetJS

SheetJS (also known as the xlsx library) is a powerful JavaScript library for reading, writing, and manipulating spreadsheet files directly in code. As advantages it works everywhere (browser + Node), require no Excel installation, and have large community and active development.

Link - https://git.sheetjs.com/sheetjs/sheetjs

XLSX – populate

XLSX - populate (officially xlsx-populate) is a JavaScript library for creating and editing Excel files programmatically - similar to SheetJS, but with a stronger focus on preserving styles, formulas, and formatting. Purpose: To generate, read, and modify .xlsx files while keeping Excel's original layout and appearance intact. Good for templated reports, invoices, and styled spreadsheets but slightly slower and heavier than SheetJS.

Link - https://github.com/dtjohnson/xlsx-populate

Common Node.js Libraries

FS

The fs module in Node.js provides an API for interacting with the file system. It is a core Node.js module, meaning no installation is required, and it allows for performing various file-related operations such as creating, reading, writing, updating, and deleting files and directories.

Path

The Node.js path module is a built-in core module that provides utilities for working with file and directory paths. It is designed to handle path manipulations consistently across different operating systems (like Windows, Linux, and macOS) by abstracting away the differences in path separators

---

CONSOLE COMMANDS:

Commands that we execute in terminal.

Commands will be separated by their functionality.

Identity, System & Configuration

- display version
- display clock
- display patch-information
- display startup
- display current-configuration
- display license esn
- display device <slot> (Used to capture power/device details)

Interfaces & Layer 2 Protocols

- display interface brief
- display interface ethernet brief
- display ip interface
- display mac-address <dynamic/summary>display vlan
- display eth-trunk
- display lacp peer

Routing & Layer 3 Protocols

- display router id
- display router id vpn-instance <name>
- display ip routing-table statistics
- display arp alldisplay arp statistics

- display vpn-instance
- display ospf brief
- display ospf peer brief
- display bgp peer
- display bgp vpnv4 peer

Specialized Protocols (MPLS, SR, BFD, VXLAN)

- display segment-routing
- display mpls lsp statistics
- display tunnel-info statistics
- display bfd session all
- display bfd configuration
- display vxlan

Management & Status

- display ntp status
- display ntp unicast-server
- display alarm
- display alarm active verbose

PROGRAM FUNCTIONS:

Functions will be grouped by their category and purpose.

Core CLI and Utility Functions

- getArg - Retrieves the value following a specified flag from command-line arguments.
- ensureOutDir - Creates the output directory if it does not already exist.out
- PathFor - Generates the output JSON file path based on the input file name.
- cleanTailPrompt - Cleans up VRP command block output by removing carriage returns and the trailing prompt.
- Uniq - Returns an array with duplicate values removed (using a Set).
- normalizeKey - Normalizes a command string to a lowercase, underscore-separated key for internal indexing.
- Lines - Splits a text block into an array of lines.
- exportFile - The primary logic function. Reads a single JSON file, initializes the Excel workbook, calls all buildSheet functions, and saves the final .xlsx file.
- createWindow - Creates and configures the main BrowserWindow, sets its dimensions and webPreferences (including the preload script), and loads the index.html file.
- setupIPC - Initializes all IPC handler listeners (ipcMain.handle). This function is called only once when the application is ready.

Log Processing and Model Helpers

- splitBlocks - Splits the raw log content into individual command blocks, either using the standard Huawei marker or a fallback heuristic based on prompt lines.
- detectCommand - Attempts to find the executed VRP command within a log block by iterating over lines and matching common command prefixes.
- newModel - Initializes and returns the complete, empty JSON structure (model) used to store all parsed data.
- ensureInterface - A utility function to retrieve an interface object from the model, creating a new entry if the interface name is not already present.

- parseFile - The main logic function. It reads the file, initializes the model, iterates through all command blocks, calls the appropriate parser function based on the command, and writes the final JSON output.

Sheet Builder Functions

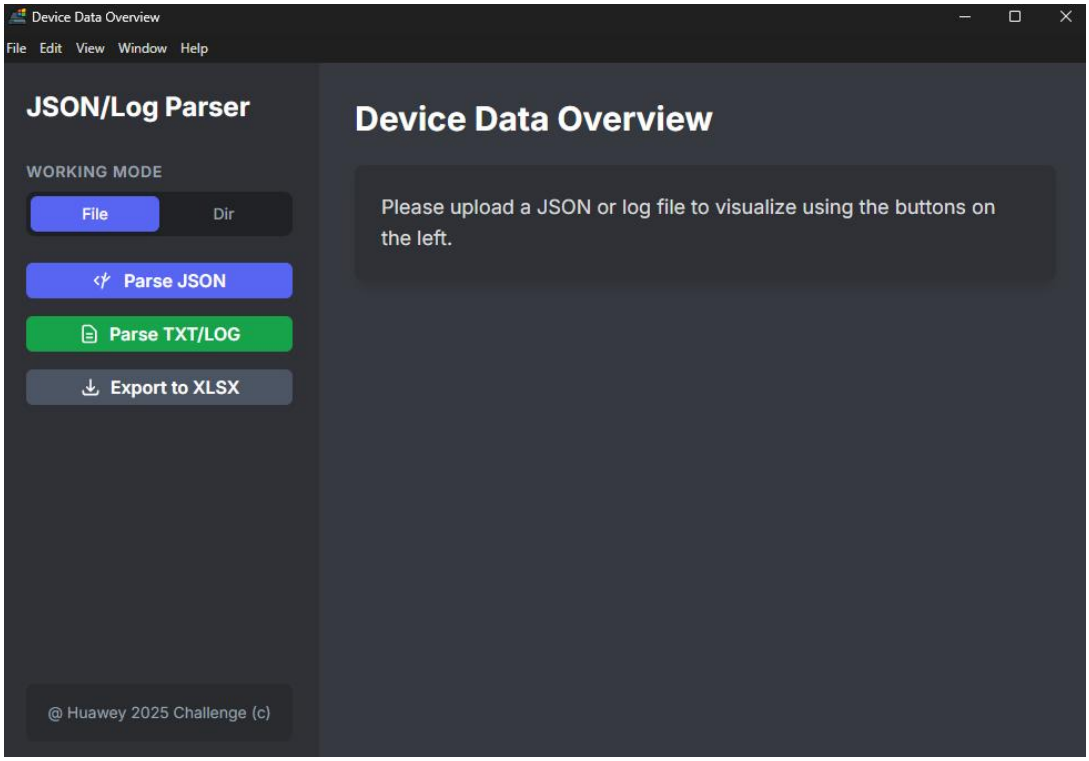
- buildSummarySheet - Creates the main identity, resource, and count overview sheet.
- buildInterfacesSheet - Populates interface details including L2 status, IP addressing, and optical/protocol fields.
- buildRoutingSheet - Lists protocol details like VRFs, BGP neighbors, and OSPF neighbors.
- buildHardwareSheet - Details hardware inventory including cards, power supplies, and SFP module metrics.
- buildAlarmsSheet - Lists all parsed alarms and license usage information.

Inter-Process Communication (IPC) Handlers

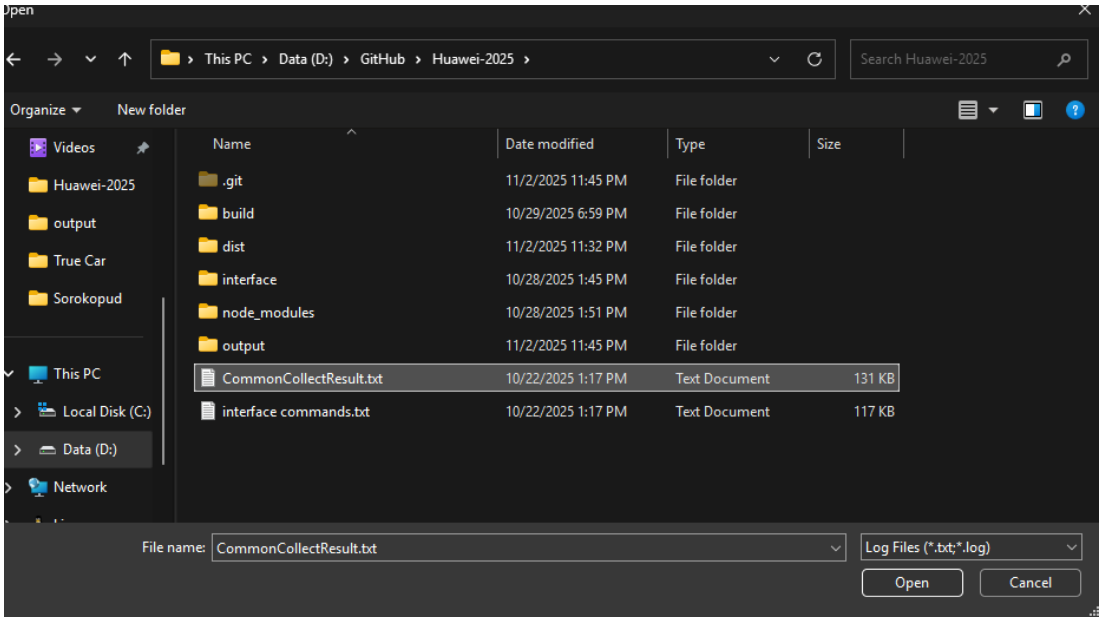
- dialog:openFileOrDirectory - Opens the native file dialog to let the user select a file, applying specific file filters (txt, log, json).
- analyze:start - Executes the imported ./analyzer.js module on the provided file path. It handles execution, logging, and returns the parsed result and output path
- file:read - Reads the content of an arbitrary file path provided by the renderer process (used primarily for reading JSON results).
- export:excel - Executes the imported ./export_to_excel.js script using a specified JSON file path to generate the final Excel workbook.

USER MANUAL

1. Run “dist/Huawei Log Analyzer 1.0.0.exe”

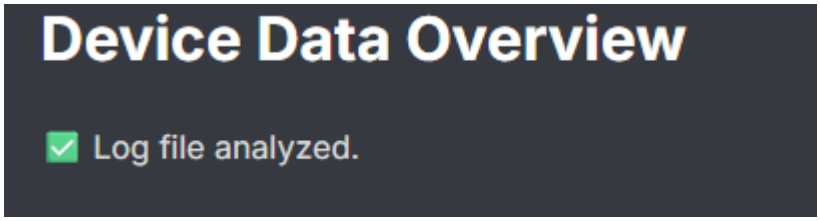


2. Choose parsing “txt\log” file or switch to “Dir” tab and choose folder to analyze all containing “txt\log” files



3. Wait until analysis over and achieved succeed message.

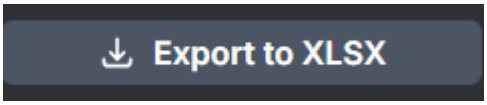
   Analyzed JSON files saving to output folder

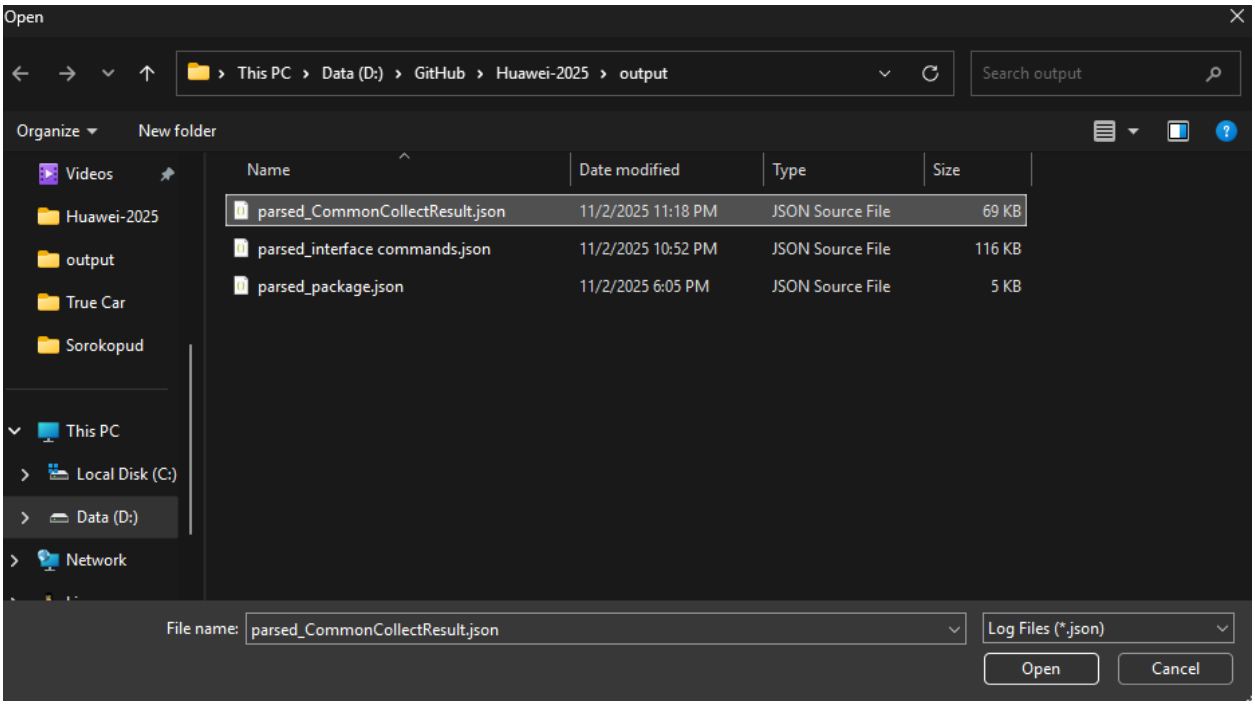


4. Now you can choose between watching info from JSON or export data to XLSX file.

   4.1) If you want to check JSON structured log data, choose “Parse JSON” and select the JSON file you want to see.

4.2) If you want to export data into XLSX, choose JSON file that you want to export, await succeed massage or switch to “Dir” tab and choose folder to analyze all containing “json” files. And XLSX file will be saved at the same directory JSON was. All output XLSX file/files will we saved to output folder





5. After you finish all work that must be done, close program by pressing red cross in the top-right corner.