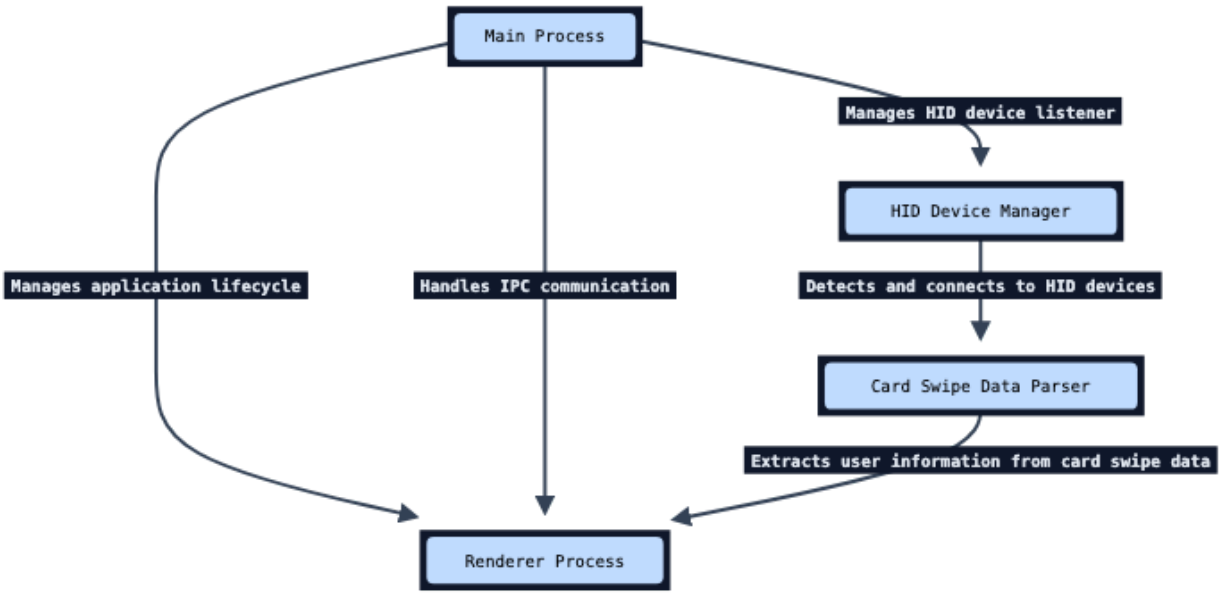
1. Onecard Swipeout

1.1 Table Of Contents

Summary



Onecard Swipeout is an Electron-based desktop application that solves the real-world problem of processing card swipe data from HID-capable magnetic stripe readers. The application's core functionality is to detect connected HID devices, extract user identification information (name and Onecard ID) from the card swipe

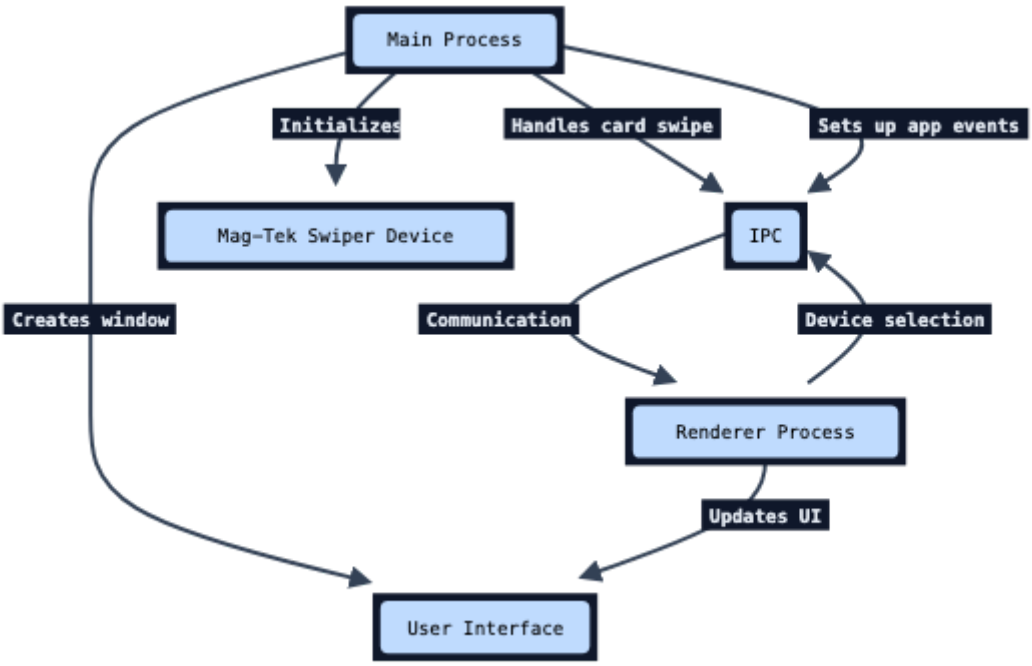
data, and display this information to the user. Additionally, the application will emulate keystrokes of the 7-digit Onecard ID from the swiped card.

The application is structured around a Main Process (`main.js`) and a Renderer Process (`index.html`), which communicate via IPC (Inter-Process Communication). The Main Process manages the application lifecycle, creates the main window, handles IPC communications, and manages the HID listener. The Renderer Process displays the user interface and handles user interactions, such as selecting a HID device and displaying swipe data.

The key functionality of the application is implemented in the magtekSwiper.js module, which is responsible for detecting available HID devices, connecting to the selected device, and parsing the swipe data to extract the user's name and Onecard ID. This module uses the node-hid library to interface with the HID devices.

The application includes several design choices to ensure security and reliability, like the lack of a non-volitile datastore, using context isolation for IPC communication, and gracefully handling errors. It also has installation and setup instructions, operational procedures, and maintenance and support information, found on the GitHub repo or in the `README.md` document.

1.2. Main Application Logic



The core functionality of the Onecard Swipeout application is managed by the Main Process and Renderer Process, which communicate via IPC (**I**nter-**P**rocess **C**ommunication).

The Main Process (main.js) is responsible for the following:

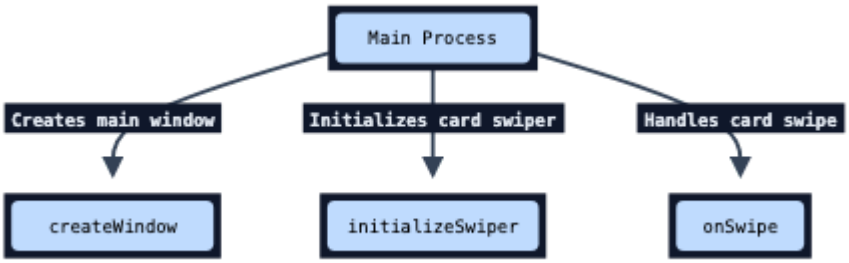
- Creating the main application window using the `Electron.createWindow()` function.
- Initializing the Mag-Tek MSR using the `initializeSwiper()` function, which calls `getMagtekSwiper()` to retrieve the HID device path.
- If multiple HID devices are detected, the Main Process sends a *select-hid* event to the Renderer Process, allowing the user to select the desired device.

- Handling the `onSwipe()` callback function, which is called whenever a card is swiped through the Mag-Tek device. If the swipe is successful, the function calls `typeString()` from the **libnut.node** native module to simulate the Onecard # being typed, and then sends a *swipe-data* event to the Renderer Process with the card data.
- Setting up event listeners for the Electron *before-quit*, *will-quit*, and *quit* events, and calling `closeSwiper()` to close the HID device when the application is about to quit.
- Handling the *activate* event to ensure that the main window is created if it was previously closed.

The Renderer Process (index.html) is responsible for:

- Listening for the *swipe-data* event, which is emitted by the Main Process when a card is swiped, and updating the UI with the card holder's name and Onecard ID.
- Listening for the *error* event, which is emitted by the Main Process when an error occurs, and updating the UI with the error message.
- Listening for the *select-hid* event, which is emitted by the Main Process when the user needs to select an HID device, and populating the HID device selection dropdown.
- Sending the selected HID device path back to the Main Process using the *hid-selection* event when the user submits their selection.
- The communication between the Main Process and Renderer Process is handled using IPC, which allows them to exchange data and control the application.

1.3. Main Process



The Main Process in `main.js` is responsible for managing the application lifecycle, creating the main window, handling IPC (Inter-Process Communication) between the Main Process and Renderer Process, and managing the HID (Human Interface Device) listener.

The `createWindow()` function sets up the main Electron application window, configuring its size, title, and application icon. This function creates a new `BrowserWindow` instance with the specified settings.

The `initializeSwiper()` function is responsible for detecting and initializing the Mag-Tek card swiper device. It calls the `getMagtekSwiper()` function to retrieve the HID device path. If multiple HID devices are detected, the function sends a *select-hid* event to the renderer process, allowing the user to select from the provided list of available HID devices.

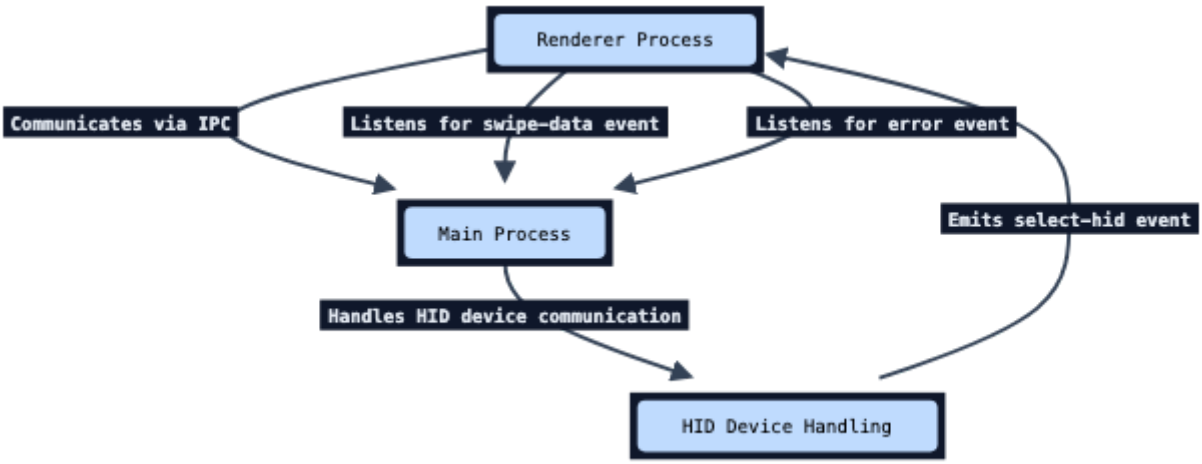
The `onSwipe()` callback function is called whenever the HID device emits a *data* event. If the swipe is successful, the function calls the `typeString()` function from the (Nut.js)[libnut.node] module to simulate the card data being typed, and then sends a *swipe-data* event to the renderer process with the card data.

The Main Process also sets up event listeners for the Electron *before-quit*, *will-quit*, and *quit* events. When the application is about to quit, the `closeSwiper()` function is called to close the HID device. This prevents

orphaned HID listener threads running after the Main Process thread is closed.

Finally, the *activate* event is handled to ensure that the main window is created if it was previously closed.

1.4. Renderer Process



The Renderer Process is responsible for displaying the user interface and handling user interactions, such as selecting a HID device and displaying swipe data. It communicates with the Main Process via Inter-Process Communication (IPC) to exchange data and control the application.

The `index.html` file is the main HTML file for the Renderer Process. It provides the user interface for the "Onecard Swipeout" application. The key functionality of the Renderer Process includes:

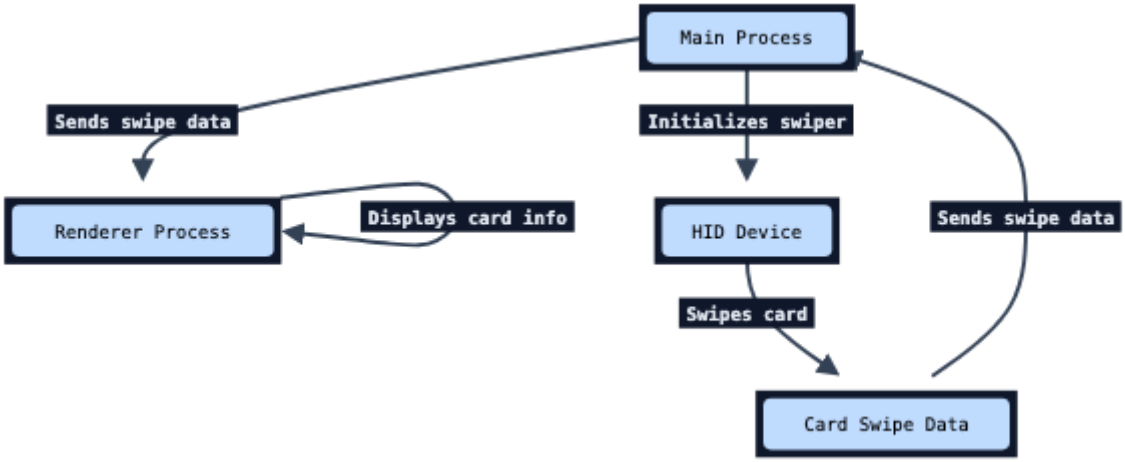
Swipe Data Handling: The Renderer Process listens for the *swipe-data* event, which is emitted by the Electron application when a card is swiped. When this event is received, the application updates the `#data` element with the card holder's name and Onecard ID.

Error Handling: The Renderer Process listens for the *error* event, which is emitted by the Electron application when an error occurs. When the error event is received, the application updates the `#data` element with the error message, displaying it in red.

HID Selection: The Renderer Process listens for the *select-hid* event, which is emitted by the Electron application when the user needs to select an HID device. When this event is received, the application populates the `#hid-selector` dropdown with the available HID devices and displays the `#hid-selector-parent` element. When the user clicks the "Submit" button, the application sends the selected HID device path back to the Electron application using the `hid-selection` event.

The Renderer Process does not define any classes or functions within the `index.html` file itself. It relies on the functionality provided by the Electron framework to handle the user interface and IPC communication.

1.5. IPC Communication



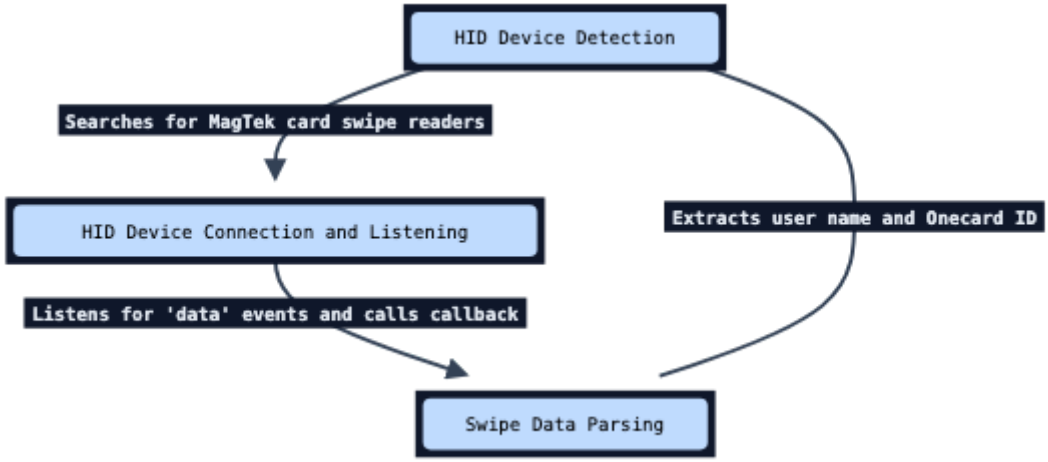
The Main Process and Renderer Process communicate via IPC (Inter-Process Communication) to exchange data and control the application.

The `main.js` file in the OnecardSwipeout directory sets up the main Electron application window and manages the communication between the Main Process and Renderer Process:

- The `initializeSwiper()` function is responsible for detecting and initializing the Mag-Tek card swiper device. When a card is swiped, the `onSwipe()` callback function is called, which sends a *swipe-data* event to the Renderer Process with the card data.
- The Main Process also listens for the *hid-selection* event from the Renderer Process, which is emitted when the user selects an HID device from the list.
- When the application is about to quit, the `closeSwiper()` function is called to close the HID device.
- The `index.html` file is the main user interface for the application and handles the communication with the Main Process:
    - The Renderer Process listens for the *swipe-data* event from the Main Process and updates the UI with the card holder's name and Onecard ID for 5 seconds, before clearning that data and returning to the "Waiting for swipe" status page. Both values are sourced from the swiped card data.
    - The Renderer Process also listens for the *select-hid* event from the Main Process, which prompts the user to select an HID device. When the user submits their selected device, the Renderer Process sends the *hid-selection* event back to the Main Process with the selected device's path.
    - The Renderer Process listens for the *error* event from the Main Process and updates the UI with any error messages.

By using IPC, the Main Process and Renderer Process can communicate and exchange data, allowing the application to handle HID device detection, card swiping, and user interface updates.

1.6. HID Device Detection and Handling



The `magtekSwiper.js` module is responsible for the core functionality of detecting, connecting to, and listening for data from HID devices, particularly MagTek card swipe readers. This module uses the **node-hid** library via the MIT License.

The key functionality of this module includes:

1.6.1. HID Device Detection:

The `getMagtekSwiper()` function searches for connected HID devices, specifically looking for devices with a vendor ID of *2049* or a manufacturer name of "Mag-Tek". This function returns the device path of the detected MagTek card swipe reader.

1.6.2. HID Device Connection and Listening:

The `startListeningToSwiper()` function takes the device path and a callback function as arguments. It creates a new `HID.HID` instance for the specified device path and starts listening for *data* events. When data is received, the function calls the provided callback with the parsed card data.

1.6.3. Swipe Data Parsing:

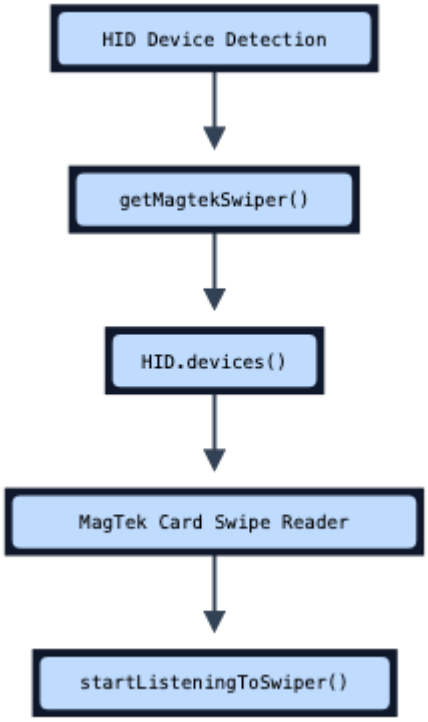
The `parseSwipeData()` function is responsible for extracting the user's name and Onecard ID from the raw card swipe data. It uses regular expressions to match the relevant information within the swipe data.

The `magtekSwiper.js` module also provides a `closeSwiper()` function to close the connection to the HID device when the application is about to quit.

The main application logic, as defined in the `main.js` file, utilizes the functionality provided by the `magtekSwiper.js` module to detect and interact with the MagTek card swipe reader. It calls the `getMagtekSwiper()` function to retrieve the device path, and then uses the `startListeningToSwiper()` function to start listening for card swipe events. When a card is swiped, the application processes the data and emits events to the renderer process to update the user interface.

For more information on the user interface and interactions, please refer to the User Interface and Interactions section.

1.7. HID Device Detection



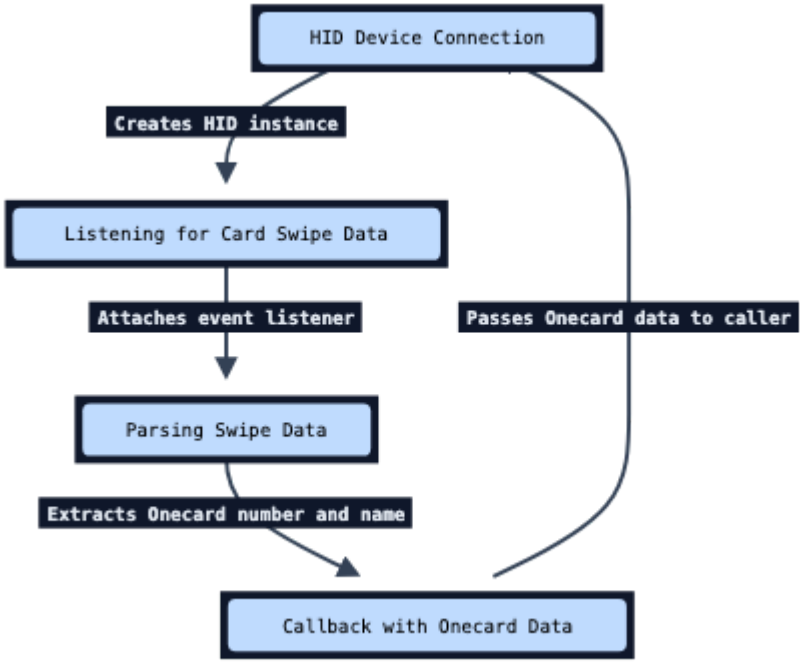
The `magtekSwiper.js` file contains the logic for detecting and connecting to HID (Human Interface Device) card swipe readers, particularly MagTek card swipe readers.

The `getMagtekSwiper()` function is responsible for detecting available HID devices and identifying any MagTek card swipe readers. It does this by:

- Retrieving a list of all HID devices connected to the system using `HID.devices()`.
- Searching the list of devices for any that have a vendor ID of 2049 or a manufacturer name of "MagTek". These are the identifying characteristics of MagTek card swipe readers.
- If a MagTek device is found, the function returns the device's path. If no MagTek device is found, it returns the paths of all available HID devices.

This function is used to obtain the device path for the card swipe reader, which is then passed to the `startListeningToSwiper()` function to begin monitoring the device for card swipe events.

1.8. HID Device Connection and Listening

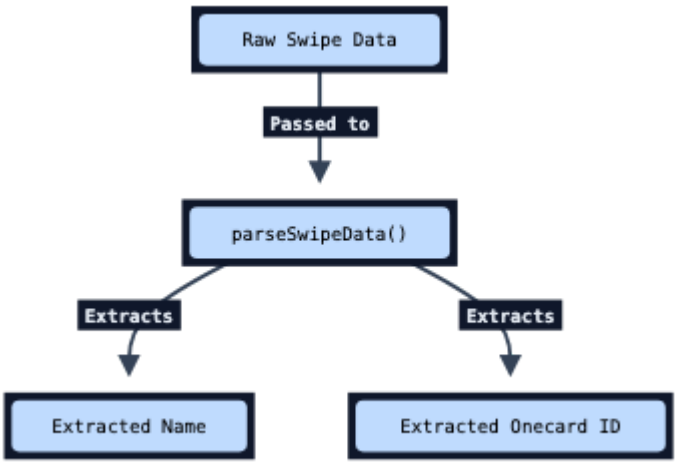


The `startListeningToSwiper()` function in `magtekSwiper.js` is responsible for connecting to the selected HID device and listening for card swipe data. This function takes two arguments: the device path and a callback function.

The key steps in the HID device connection and listening process are:

- The function creates a new `HID.HID` instance using the provided device path.
- It then attaches an event listener for the *data* event on the HID device.
- When *data* is received from the device, the event listener calls the provided callback function, passing along the raw swipe data.
- Inside the callback, the `parseSwipeData()` function is called to extract the Onecard number and name from the raw swipe data.
- The extracted Onecard data is then passed back to the caller via the callback function.
- The `closeSwiper()` function is used to close the connection to the HID device when it is no longer needed.

1.9. Swipe Data Parsing



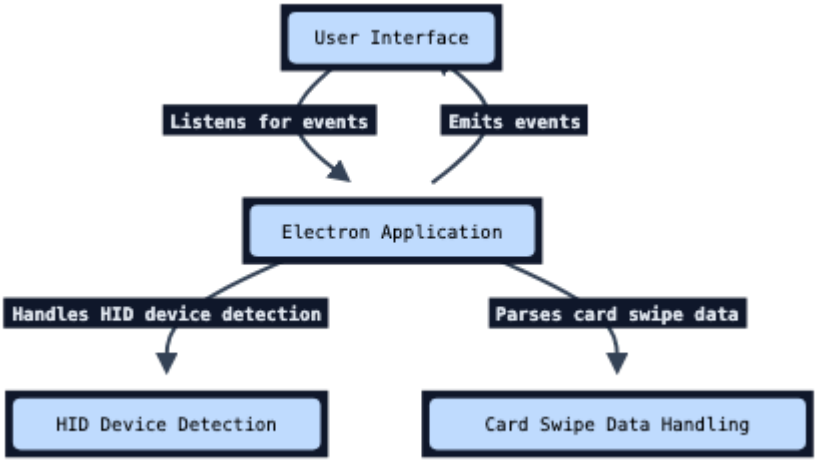
The `parseSwipeData()` function in `magtekSwiper.js` is responsible for extracting the user's name and Onecard ID from the raw card swipe data.

The function first removes any periods (.) from the input data using the `replace()` method. It then uses a regular expression to find the name, which is the text between the ^ characters in the swipe data. The function trims any leading or trailing whitespace from the extracted name.

Next, the function uses another regular expression to find the 7-digit Onecard ID, which is the digits followed by 3 spaces in the swipe data. The function returns an object containing the extracted name and Onecard ID.

If the function is unable to find the name or Onecard ID in the input data, it throws an *error* with a message indicating that the Onecard and name could not be determined.

1.10. User Interface and Interactions



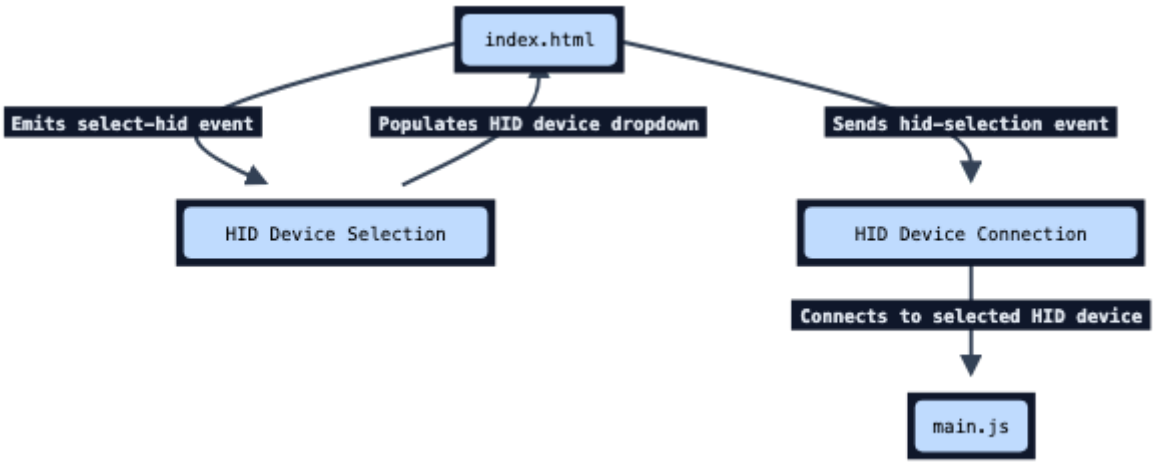
The user interface and interactions of the "Onecard Swipeout" application are primarily handled in the `index.html` file. This file provides the HTML structure and event handling for the application's graphical user interface (GUI).

The key functionality of the user interface includes:

- Swipe Data Handling: The application listens for the *swipe-data* event, which is emitted by the Electron application when a card is swiped. When this event is received, the application updates the `#data` element with the card holder's name and Onecard ID.
- Error Handling: The application listens for the *error* event, which is emitted by the Electron application when an error occurs. When the error event is received, the application updates the `#data` element with the error message, displaying it in red.
- HID Device Selection: The application listens for the *select-hid* event, which is emitted by the Electron application when the user needs to select an HID device. When this event is received, the application populates the `#hid-selector` dropdown with the available HID devices and displays the `#hid-selector-parent` element. When the user clicks the "Submit" button, the application sends the selected HID device path back to the Electron application using the `hid-selection` event.

The user interface is designed to provide a simple and intuitive way for users to interact with the "Onecard Swipeout" application. The application's main functionality, such as detecting and connecting to HID devices, parsing card swipe data, and emulating keystrokes, is handled by the Electron application's main process and the magtekSwiper.js module, as described in the Main Application Logic and HID Device Detection and Handling sections.

1.11. Selecting HID Device



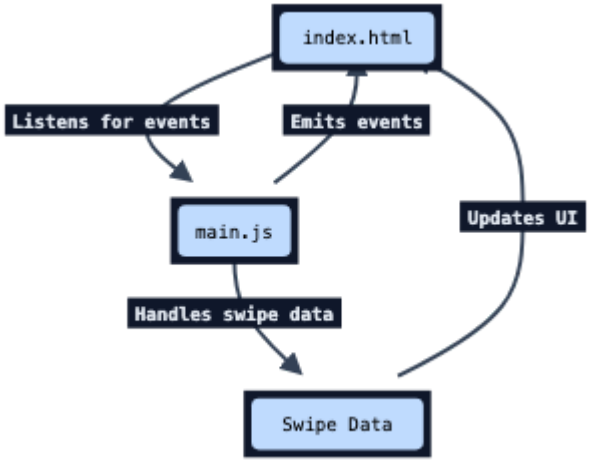
The `index.html` file provides the user interface for the "Onecard Swipeout" application, including the functionality for allowing the user to select a HID device.

When the application needs the user to select an HID device, it emits the *select-hid* event. This event is listened for in the `index.html` file, which then populates a dropdown (`#hid-selector`) with the available HID devices.

The available HID devices are provided to the `index.html` file by the Electron application. When the user selects a device from the dropdown and clicks the "Submit" button, the `index.html` file sends the selected HID device path back to the Electron application using the *hid-selection* event.

This allows the Electron application to connect to the selected HID device, such as a MagTek card swipe reader, and start listening for card swipe data.

1.12. Displaying Swipe Data



The `index.html` file in the OnecardSwipeout directory handles the display of card swipe data in the "Onecard Swipeout" application.

When a card is successfully swiped, the application listens for the *swipe-data* event, which is emitted by the Electron application. When this event is received, the application updates the *#data* element on the page with the card holder's name and Onecard ID.

If an error occurs during the card swipe process, the application listens for the *error* event, which is also emitted by the Electron application. When the *error* event is received, the application updates the *#data* element with the error message, displaying it in red.

1.13. Glossary

- **MSR**: Magnetic Stripe Reader - A device designed to read data encoded on a magnetic stripe, typically on a card used for identification or access control.
- **HID**: Human Interface Device - A device standard designed to replace older proprietary connectors with a generic USB driver for most basic tools used to interact with an electronic system. Examples would include: alphanumeric displays, bar code readers, speakers, headsets, auxiliary displays, sensors, and many others.
    - Note: Not all MSRs are HID-capable. The lower cost options are typically keyboard emulators, but aren't HID-compliant and will not be compatible with this software as a result. (MSR90 is an example of such an incompatible device.)

Doc: Comprehensive Examination of Onecard Swipeout *Author*: *Dave Luhman, Founder, ADO Software Date*: 08-29-2024 *Revision*: 1.0