SDK API Methods

The integrator needs the following APIs of the **IDEMIA Verify SDK - Linux** for communication with the **IDEMIA Mobile ID App**.

In this example, the **IDEMIA Mobile ID App** is an Android app and works as the **Mobile ID** credential holder.

The header file *Terminal.h* acts as the entry point for initializing the device engagement, sending a request to the **Mobile ID** credential holder, receiving the response, and sending the parsed model back to the calling app.

The *Terminal.h* file exposes the following APIs to caller/client with the help of a concrete class terminal:

- `initDeviceEngagement`

  This API initializes the device engagement and the security model using the configuration provided by the end-user. It then returns an error or a `Device_Engagement` model `on_Failure` or `on_Success` respectively via the `Callback` function.

```
static void initDeviceEngagement(IConfigBuilderDevEng*, IResponseCallback*);
```

- `sendRequest`

  This API establishes the connection with the **Mobile ID** backend services and requests the **Mobile ID** credential fields using the configuration provided by the **IDEMIA Mobile ID Verify App**. When it's successful, it returns the `MDL_Data` model, along with other values through the `Callback` function.

```
static void sendRequest(IConfigBuilderSendRequest*, IResponseCallbackMdl* );
```

- `cancelRequest`

  This API cancels the ongoing request made to the **IDEMIA Mobile ID App** by disconnecting from the BLE/NFC/WifiAware and cleans all the request data.

```
static void cancelRequest();
```

- `parsePDF417`

In this API, set the PDF417 barcode data through the `IConfigBuilderDevEng` object. When successful, it returns the `MDL_Data` model containing the values which were present in the barcode through the `Callback` object.

```
static void parsePDF417(IConfigBuilderDevEng*, IResponseCallbackMdl*);
```

- `Version`

  This API returns the version of the **IDEMIA Verify SDK - Linux**.

```
static void version();
```

- `enableDebug`
  This API enables Debug Mode. With this, extra debug logs get printed on the console and also gets dumped in a log file inside the integrator's application.

```
    static void enableDebug();
```