Payment Device SDK for Android

Axium Development Guide Supplement for Release 5.4.2 (Cassiopeia)

NMI
© NMI. All rights reserved.

Table of contents

1. Introduction 4
2. Setup for Development 4
- Device Modes 4
  - Development 4
  - Production 6
- Pre-Requisites 6
- Integration Considerations 6
  - Kiosk Mode 6
  - Android Library Restrictions 6
  - Connection 7
  - UI Updates 7
3. Kiosk Mode for Your Application 9
- What Kiosk Mode Covers 9
- Becoming the Home App 10
- Blocking the Notification Panel 11
- Opening the Admin Screen 12
- Coming Back After a Payment 12
- Updates and Restarts 13
- Known Limitations 13
4. HEM 13
- HEM Concepts 14
  - Users and Roles 14
  - Campaigns 14
  - Region 14
  - Sponsors 14
  - Estates 15
  - Terminals 15
  - Call Schedule 15
- HEM Navigation 16
  - Software Library 16
  - Campaigns 17
  - Device Management 17
  - MDM Profiles 17
  - User Management 17
  - Sponsor Selection 19

Requesting Access 19
Device Onboarding Process 20
Partner Onboarding with a Partner Code 21
Setting Up for Partner Code Onboarding 22
The End-User Experience 22
Role Access Levels 23
PGP-ADMIN 23
PGP-CAMPAIGN-MANAGER 23
PGP-DEVICE-MANAGER 23
PGP-MDM-MANAGER 24
PGP-VIEW-ONLY 24
Admin Controls 24
User Management 24
Create a New Account 25
Defining Access 26
Device Manager Controls 28
Managing Estates 28
Adding Estates 28
Adding Terminal 30
Call Schedule 31
Campaign Managers 32
Upload Software 32
Campaign Creation 32
MDM Managers 34
Viewers 36
5. Application Signing 36
Signing 37

Axium Development Guide Supplement

Android supplement for Payment Device SDK Release 5.4.2 (Cassiopeia).

1. Introduction

This supplement provides information for integration partners who wish to develop their own applications to run on Axium devices. It covers the steps required to set up development on an Axium device, as well as how to access and use Ingenico's Hosted Estate Manager (HEM) and E-Signing platform.

2. Setup for Development

Development Axium devices operate just like a normal Android device — once plugged into your development machine they will appear in Android Studio. You will also use the ChipDNA Mobile SDK as you would in a standard semi-integrated environment.

Device Modes

When selecting a device it's important to get the correct configuration for the purpose you require it for. Axium devices come in two variants.

Development

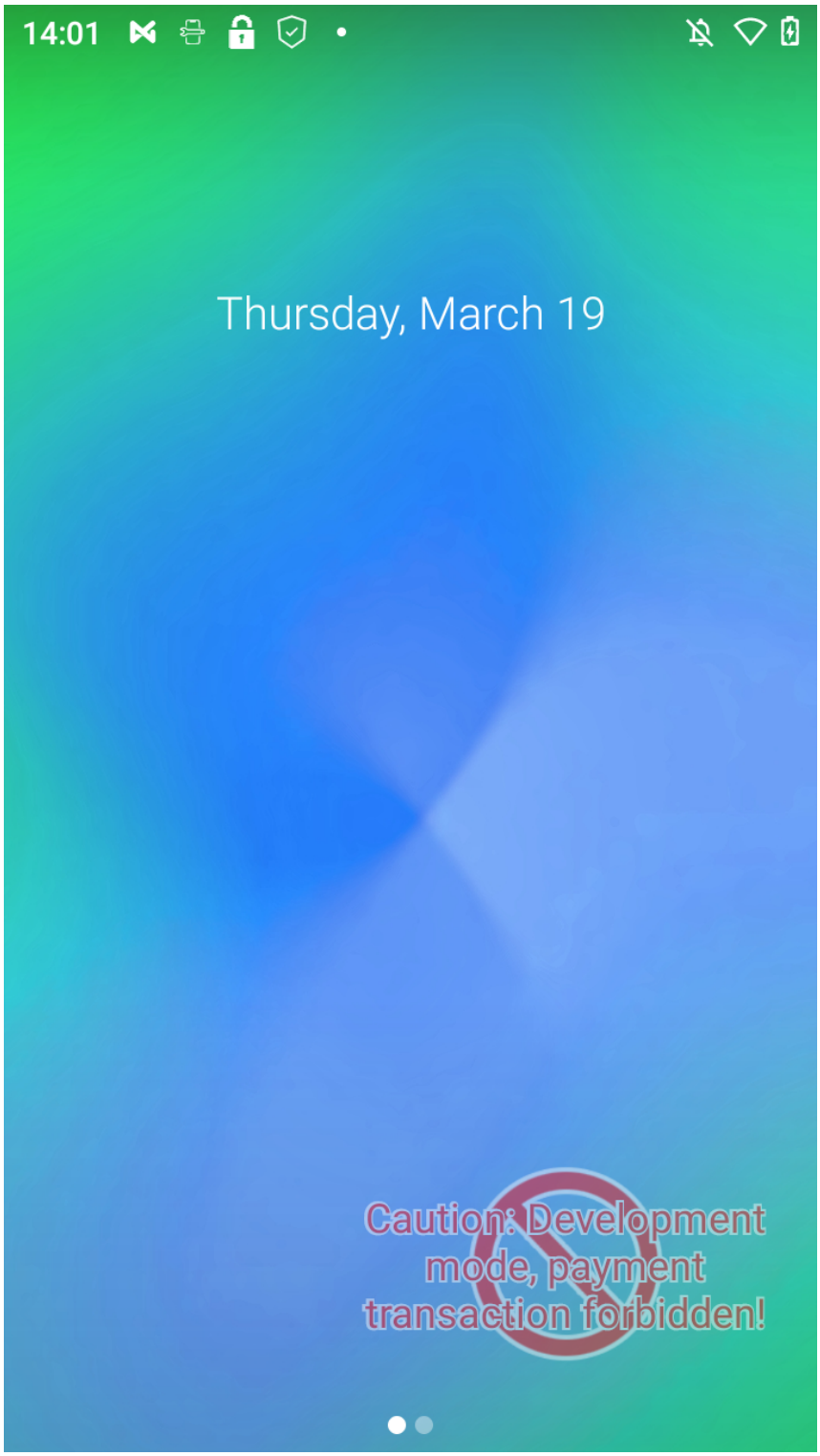
Development devices are injected with test keys and can only be used in the Test environment. They are intended for building and testing applications directly on the device.

These devices:

- Ignore application signature validation
- Can be connected to Android Studio
- Display a warning message on the home screen

Once a device has been set to Development mode, it cannot be reverted to Production mode without being returned to Ingenico.

Development devices can be identified by a flashing warning sign on the home screen.



Production

Production devices are injected with production keys and can be used in both Test and Production environments. These are intended for standard usage where no on-device development is required.

These devices:

- Only run applications signed via Ingenico's e-signing portal
- Do not appear in Android Studio
- Cannot be used for development

Pre-Requisites

- Android SDK Target 29
- Android Studio
- ChipDNA Mobile SDK v5.2.0 or above

Integration Considerations

Developing on a development Axium device is very similar to developing an application for standard Android. However there are some considerations to think about.

Kiosk Mode

The Transaction Initiator UI has a kiosk mode of its own, which keeps it in front whenever no transaction is running, so another app on the same device cannot stay in the foreground. Do not turn it on if your point-of-sale app runs on the Axium device itself. To lock the device to your own app instead, see Kiosk Mode.

Android Library Restrictions

While the version of Android running on Axium devices is based on the standard Android distribution, some standard Android libraries have been removed in order for the OS to meet PCI security requirements.

Connection

In order to connect to the TransactionInitiatorUI application from an application running on the same device you must use the following connection properties:

- Connection Type: TCP-IP
- IP Address: `0.0.0.0`
- Port No: `8088`

To set this within the ChipDNA Mobile SDK it would look like the following:

```
Parameters requestParameters = new Parameters();
requestParameters.add(ParameterKeys.PinPadConnectionType,
                      ParameterValues.TcpIpConnectionType);
requestParameters.add(ParameterKeys.PinPadName, "Axium-" + Build.MODEL);
requestParameters.add(ParameterKeys.PinPadIpAddress, "0.0.0.0");
requestParameters.add(ParameterKeys.PinPadPort, "8088");

ChipDnaMobile.getInstance().setProperties(requestParameters);
```

UI Updates

When making calls to the ChipDNA Mobile SDK, the Ingenico ARC UX application is brought to the foreground to handle transaction processing. This behaviour is controlled by Ingenico, and NMI cannot return the integrating application to the foreground automatically.

While the ARC UX application is in the foreground, your application continues to run in the background and will still receive standard updates from the ChipDNA Mobile SDK.

If your application needs to regain control and return to the foreground, call the method below from your transaction listener — typically at the end of `onTransactionFinishedListener` (and, if required, on the `GetCardDetailsCompleted` transaction update).

Two points to note:

- Use an `Intent` with `FLAG_ACTIVITY_REORDER_TO_FRONT` as the primary approach. `moveTaskToFront` is increasingly restricted on modern Android and is used here only as a fallback.
- A short delay is recommended before bringing your application forward, so the ARC UX application can finish its own screen teardown first.

The method must be a member of the `Activity` you want to bring to the foreground, so that `this` and `taskId` resolve correctly:

```
/**
 * Returns the integrating application to the foreground after a transaction
 * completes. Call from your transaction listener (e.g. onTransactionFinishedListener).
 *
 * @param delayMs delay before returning to the foreground, allowing the ARC UX
 *                application to finish its screen teardown first.
 */
private fun bringAppToForeground(delayMs: Long = 500) {
    Handler(Looper.getMainLooper()).postDelayed({
        try {
            // Preferred: re-order this app's existing task to the front.
            val intent = Intent(this, YourActivity::class.java).apply {
                addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT or
                         Intent.FLAG_ACTIVITY_SINGLE_TOP)
            }
            startActivity(intent)
        } catch (e: Exception) {
            // Fallback: move the task to the front via ActivityManager.
            val activityManager =
                getSystemService(Context.ACTIVITY_SERVICE) as ActivityManager
            activityManager.moveTaskToFront(
                taskId, ActivityManager.MOVE_TASK_NO_USER_ACTION)
        }
    }, delayMs)
}
```

Call it from your transaction-finished handler:

```
override fun onTransactionFinishedListener(parameters: Parameters) {
    // ... your existing transaction handling ...
    bringAppToForeground()
}
```

The following permission will also need to be added to the application's manifest file:

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
```



From Android API 29, an application in the background is prevented from launching activities or moving its task to the foreground. The `SYSTEM_ALERT_WINDOW` permission exempts your application from this restriction, which is what allows the code above to work. On Axium devices the permission is granted when your application is installed, so there is nothing to grant on the device. You can check it at runtime with `Settings.canDrawOverlays(context)`; if it returns `false`, the calls above will silently have no effect.

3. Kiosk Mode for Your Application

In an All-in-One deployment your application, not the Transaction Initiator UI, is the one staff use all day. To lock the device to it, build kiosk mode into your application with standard Android APIs, and use the ChipDNA Mobile SDK for what an application cannot do itself: taking payments, and blocking the notification panel.

Leave the Transaction Initiator UI's own kiosk mode off. Never pass `ParameterKeys.KioskModeEnabled` from your application: it makes the Transaction Initiator UI the Home app, in place of yours.

What Kiosk Mode Covers

Staff action	With kiosk mode on	How
Press Home	Comes back to your application	Your application is the Home app
Pull down the notification panel	Blocked for the whole device	The SDK's `NotificationPanelAllowed` setting
Update your application, or restart the device	Kiosk mode stays on	The alias and the Home choice are kept

An application cannot make itself the Home app silently: someone has to accept Android's prompt on the device each time kiosk mode is turned on. The Recents screen, Android's list of recently used applications that opens from the Recents button, cannot be blocked by an application; see Known Limitations.

Becoming the Home App

Declare an `activity-alias` with the `HOME` category, switched off, that points at your main activity. Make the activity `singleTask`, so that Home brings back the screen that is already open:

```
<activity
    android:name=".MainActivity"
    android:exported="true"
    android:launchMode="singleTask">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity-alias
    android:name=".KioskHomeActivity"
    android:enabled="false"
    android:exported="true"
    android:targetActivity=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.HOME" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity-alias>
```

To turn kiosk mode on, switch the alias on, then ask to become the default Home app:

```
val homeAlias = ComponentName(context, "com.example.yourapp.KioskHomeActivity")

// Without DONT_KILL_APP, Android stops your application's process.
packageManager.setComponentEnabledSetting(
    homeAlias,
    PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
    PackageManager.DONT_KILL_APP
)

val roleManager = getSystemService(RoleManager::class.java)
if (!roleManager.isRoleHeld(RoleManager.ROLE_HOME)) {
    requestHomeRole.launch(roleManager.createRequestRoleIntent(RoleManager.ROLE_HOME))
}
```

`requestHomeRole` is an `ActivityResultLauncher` registered as a property of the activity. In Android's prompt, choose your application, then **Set as default**.

To turn kiosk mode off, switch the alias off. Your application is then no longer a Home app, and Home goes to the device's other Home app, normally the Android launcher:

```
packageManager.setComponentEnabledSetting(
    homeAlias,
    PackageManager.COMPONENT_ENABLED_STATE_DISABLED,
    PackageManager.DONT_KILL_APP
)
```

Switching the alias off also closes any screen that was opened through it, so if staff should stay in your application, start your main activity again afterwards.

Write the alias's class name out in full: your namespace followed by `.KioskHomeActivity`. `context.packageName` is the application ID, which can differ from the namespace.

Blocking the Notification Panel

An application cannot stop the notification panel being pulled down, which would give staff the quick settings and, from there, Android's settings. Have the Transaction Initiator UI block it instead, through `connectAndConfigure`:

```
ChipDnaMobile.getInstance().connectAndConfigure(Parameters().apply {
    add(ParameterKeys.NotificationPanelAllowed, ParameterValues.FALSE)
})
```

ARC then blocks the panel for the whole device, whether or not your kiosk mode is on. Pass `ParameterValues.TRUE` to allow it again.

This needs Transaction Initiator UI v1.4.0 or later, and ChipDNA Mobile SDK 5.4.2 or later, which added `NotificationPanelAllowed`.

- `Result` = `TRUE` in `onConnectAndConfigureFinished` means the Transaction Initiator UI has taken the setting. Normally ARC has already applied it by then. If the terminal is busy, for example during an update or just after it starts, the setting is applied as soon as the terminal is free, without another call; until then `getStatus` returns `ParameterKeys.KioskSettingsApplied` = `FALSE`.
- Send the setting again after each successful connect, in case the Transaction Initiator UI has restarted, but only a value your application has actually set. Sending a default overwrites the setting made on the Transaction Initiator UI's Admin screen. Do not send back the `NotificationPanelAllowed` that `getStatus` reports either: it is whether the panel is allowed right now, not the stored setting.
- A change made later on the Transaction Initiator UI's Admin screen replaces your setting, and then applies only while the Transaction Initiator UI's kiosk mode is on: with it off, the panel is allowed again. Send your setting again to block it.
- `AdminSettingsNotSupported` and `NotificationPanelAllowedInvalid` will not succeed on a retry. See the Axium Smart POS Supplement for the full list of errors.



The default Admin screen passcode, `141309`, must be changed. Every terminal starts with it, and the Admin screen can open Android's settings, so anyone who knows it can get out of kiosk mode. When `getStatus` returns `ParameterKeys.AdminPasscodeIsDefault` = `TRUE`, set your own passcode with `ParameterKeys.NewAdminPasscode` on `connectAndConfigure`. The Admin screen also asks for a new passcode the first time it is opened with the default.

See the Axium Smart POS Supplement for the passcode rules.

Opening the Admin Screen

The Transaction Initiator UI's Admin screen opens from its welcome screen: tap the top-left corner of the screen five times in quick succession, each tap within 0.7 seconds of the one before. It only opens while no payment or update is running. Nothing on screen marks the corner, so cardholders have nothing to find. The passcode screen opens first; on a terminal still using the default passcode, the Admin screen then asks for a new one.

Coming Back After a Payment

ARC's payment screen stays in front once the payment ends. Bring your application back to the foreground from `onTransactionFinishedListener`, as described in UI Updates.

Updates and Restarts

Android keeps the alias switched on, and your application as the default Home app, across an update of your application and a restart of the device. There is no need to turn kiosk mode off before an update through HEM, or to switch the alias on again afterwards.

After a restart, ARC's start-up screen can be in front of your application until Home is pressed. Do not start your own activity over it from a `BOOT_COMPLETED` receiver: ARC must finish starting before the Transaction Initiator UI can use it.

Known Limitations

- The Recents screen cannot be blocked by an application. Staff can open it and switch to another application that is already running; pressing Home brings your application back.
- If the Transaction Initiator UI stops answering in the middle of a payment, the SDK keeps asking it for the result and `onTransactionFinishedListener` is never called. Over the local TCP connection, `terminateTransaction` cannot end the payment either. Restart your application to recover. The payment may have been approved, so before staff take it again, call `getTransactionInformation` with the payment's `UserReference` once the SDK has connected, to see its outcome. Store each payment's `UserReference` somewhere that survives a restart, such as `SharedPreferences`, until the payment finishes.

4. HEM



This section provides a brief overview of the structure and key concepts used in HEM.

HEM is an Ingenico-owned product and may be subject to change without prior notice to NMI. The information provided here is intended as a starting point, covering basic concepts and common tasks.

For more detailed guidance, refer to Ingenico's official documentation, which is available within HEM via the navigation menu.

HEM Concepts

Users and Roles

A user is a member of your organisation that you wish to give access to HEM. `PGP-ADMIN` is the only role allowed to add new users.

Roles define the level of access a user is granted. A user can hold multiple roles, so one user can be a `PGP-DEVICE-MANAGER` and a `PGP-CAMPAIGN-MANAGER` to allow a single user to manage the devices and campaigns for those devices.

Campaigns



Campaigns allow you to assign Axium software from the software library to devices, ensuring it is installed the next time the device connects to HEM. Within a campaign you can assign software to either individual devices or entire estates, giving you flexibility over software deployment.

Region

NMI operates at the Region level of HEM. All sponsors, estates and terminals reside within NMI's region.

Sponsors

Sponsors are the highest level structure accessible for partners on HEM. Sponsors are linked to a Partner ID. Depending on your integration type this will either be:

- CardEase Client ID
- Omni Platform Affiliate ID

Each sponsor holds its own software library, estates and terminals.

Devices will be assigned to your sponsor the first time they reach out to the NMI backend. Once a device is registered on HEM it can move between sponsors by an admin or user holding the `PGP-DEVICE-MANAGER` role.

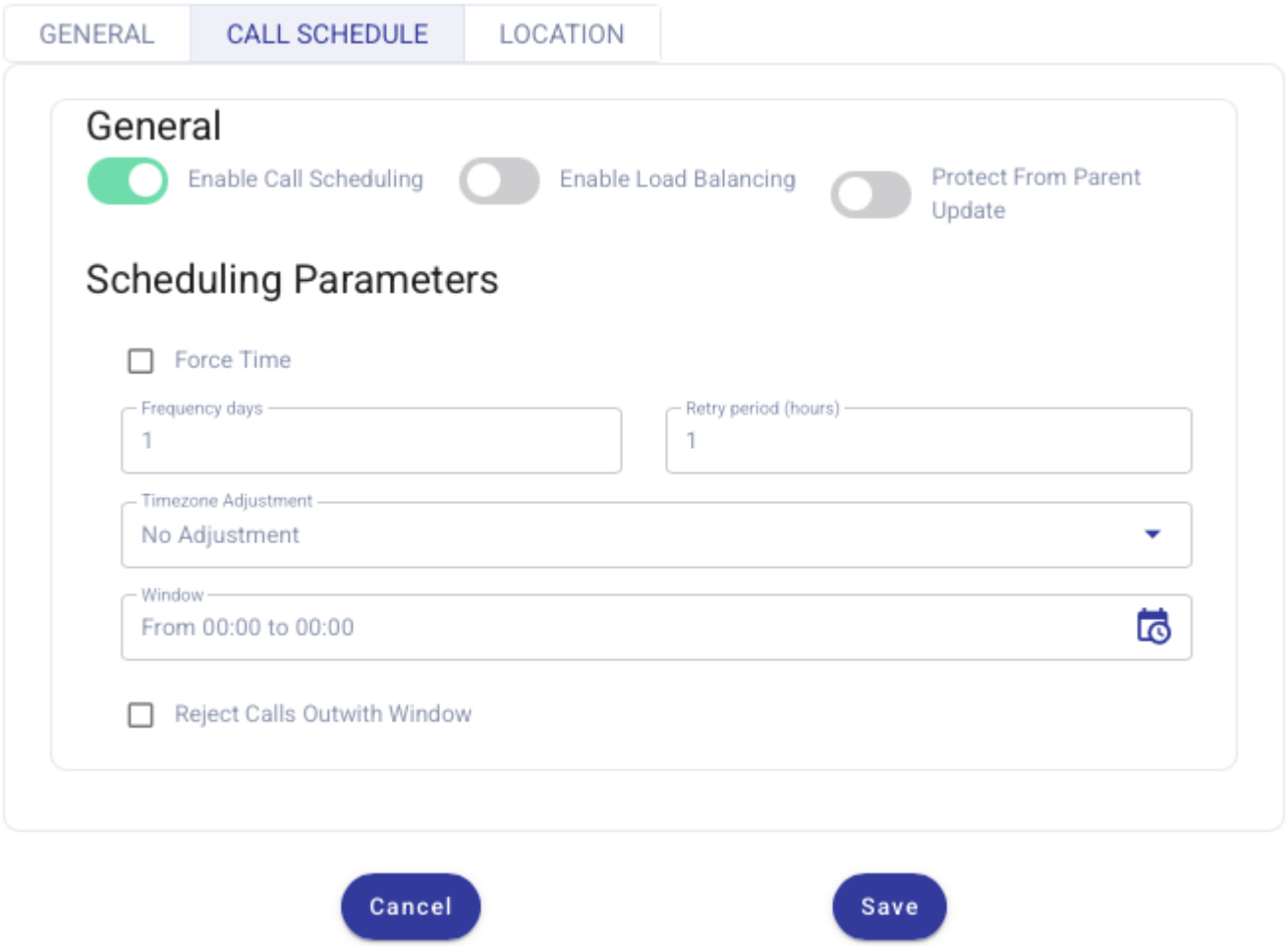
Estates

Estates can contain sub-estates and terminals. Estates are a way to group terminals. You can create as many estates as needed and organise your terminals within the estate as required. Estates enable you to assign campaigns to groups of terminals instead of needing to select terminals individually.

Terminals

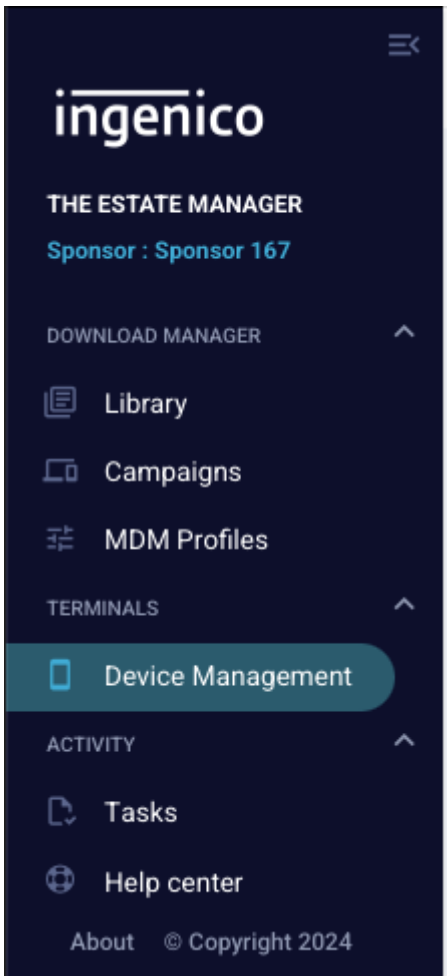
Terminals align with devices. Devices are onboarded onto HEM at the distributor and sit in a holding estate until they connect to the NMI backend. A device will be added to a sponsor's default estate when it first connects up to the NMI backend.

Call Schedule



A call schedule can be assigned to a terminal or an estate then synced to all terminals below it. By default all devices are assigned a 24 hour call schedule, meaning every 24 hours the device will make a call to HEM and run any campaigns assigned to it.

HEM Navigation



Once access is granted the left hand bar of HEM will show the navigation options. The appearance may vary depending on your access level, as HEM only displays the options available to you.

Software Library



The library section allows management of your software library. It allows you to upload your Axium-compatible software.

This must be in APK or UNS format and signed with a valid Ingenico e-signing card if you are intending to load it onto production devices.

Campaigns

The campaign section allows management of campaigns. Campaigns allow you to set software uploaded to the software library to be installed onto devices within your estate.

Device Management



From Device Management you can view all estates and terminals associated with your current sponsor. From this section you can move your devices between estates and configure call schedules.

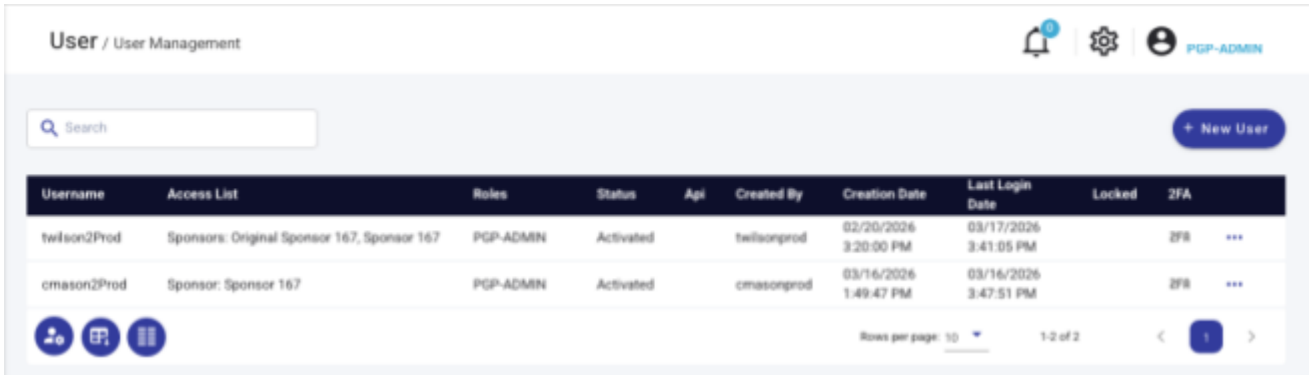
MDM Profiles

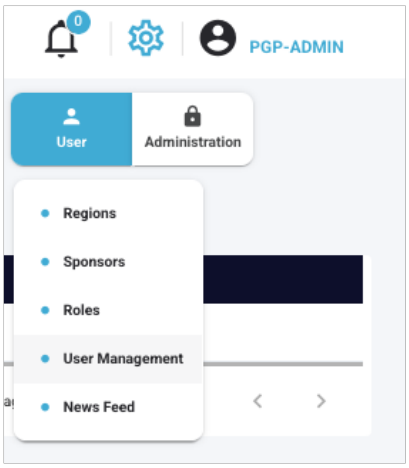


The MDM Profiles section allows you to create and manage your MDM profiles. This section allows you to control aspects of your device's OS settings such as Wi-Fi network settings, boot launching of an application, etc.

User Management

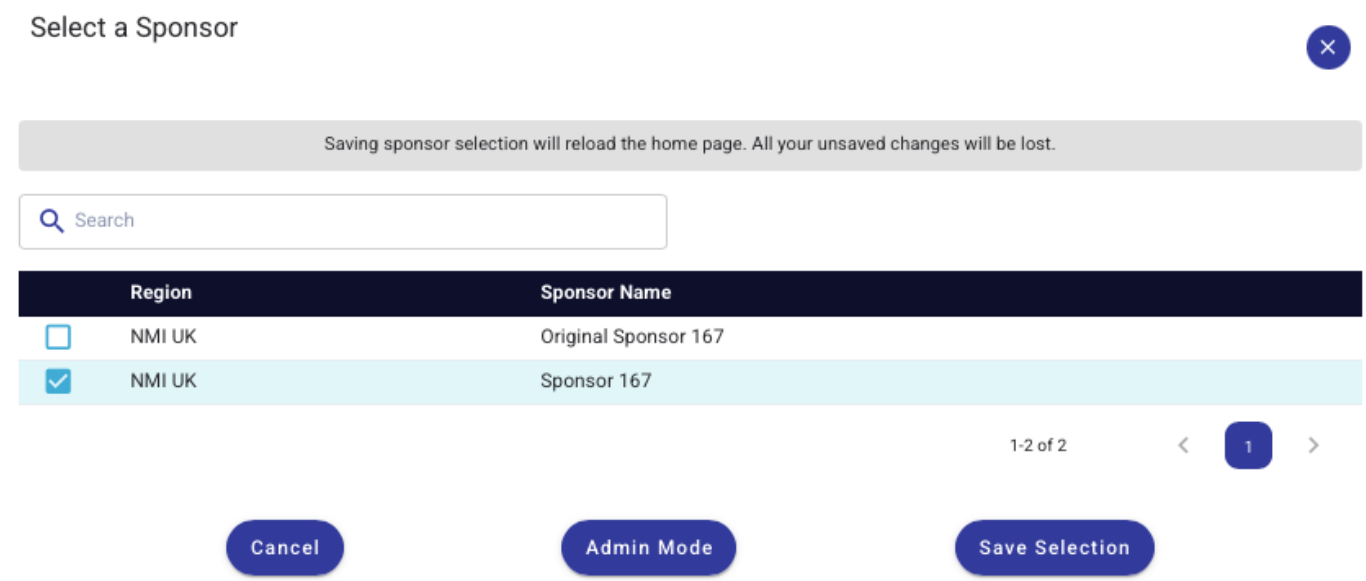






User management is available under the cog wheel icon at the top right of the page. Only users with the role `PGP-ADMIN` have access to this section.

Sponsor Selection





If your user account has access to multiple sponsors you can select which one you are viewing by clicking on the current sponsor name.

Requesting Access

Access to HEM incurs an additional cost. To request access, please contact your Account Manager.

To ensure the correct level of access is provisioned, include the following information in your request:

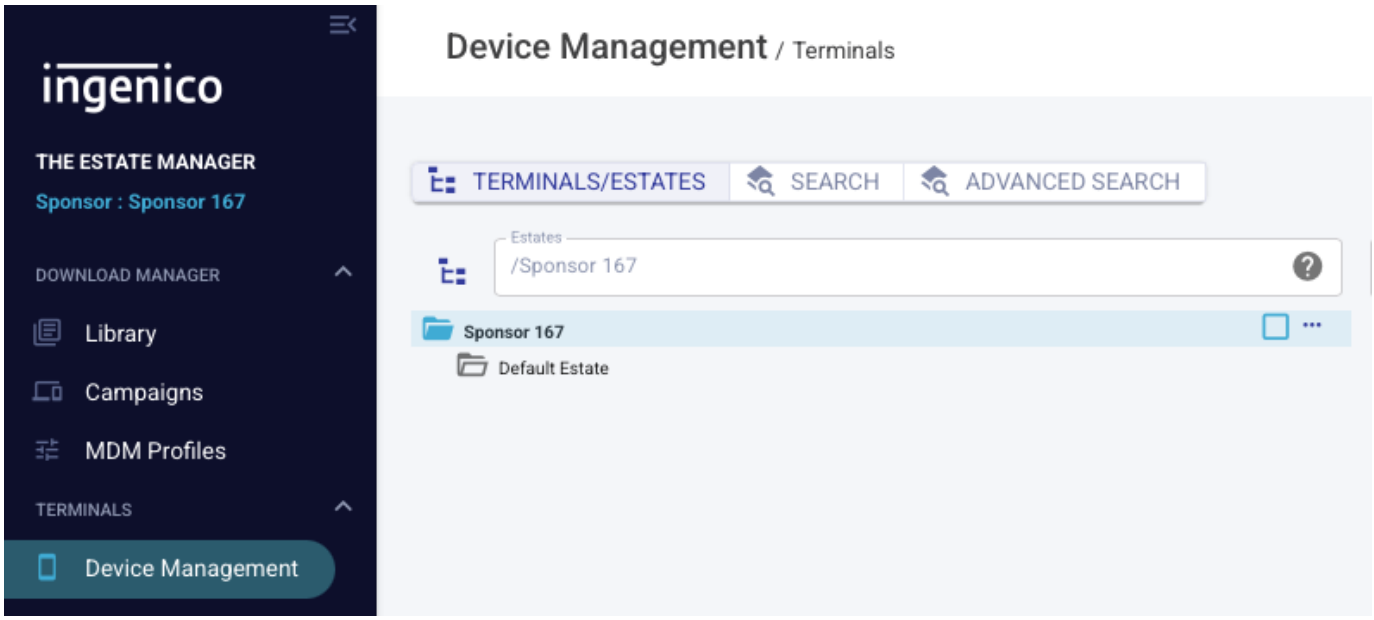
- Account Name
- Client/Affiliate ID(s) (if known), otherwise the serial number of a registered device
- Initial Admin details:
- Email
- First name
- Last name
- E-signing Portal account details (for Pushers and Validators):
- Email
- First name
- Last name

Once provisioned, an email will be sent to the provided address to complete setup of the admin account. The admin user can then grant access to additional users as required.

Device Onboarding Process

All terminals are boarded onto HEM at the point of distribution. Until a device is first used, it remains in a holding estate that is only accessible to NMI personnel. Attempting to add a device that has already been boarded onto HEM will fail.

When the device is first registered and configured with the NMI backend, it is automatically moved to a default estate under a sponsor associated with the NMI account using the device.



A sponsor is linked to an NMI partner ID, which will be one of the following:

- CardEase Client ID
- Omni Platform Affiliate ID

If you use devices across multiple Client or Affiliate IDs, you will need access to each corresponding sponsor.

Partner Onboarding with a Partner Code

By default devices are automatically moved to your sponsor's default estate the first time they register with the NMI backend based on their Client or Affiliate ID (see Device Onboarding Process above).

The Partner Code flow provides an alternative: it allows you to direct a device into a specific onboarding estate that you control. This ensures that your custom applications and configurations install automatically, without requiring the device to be tied to your specific Client or Affiliate ID.

At a high level, you prepare an onboarding estate and a permanent campaign in HEM, NMI links a Partner Code to that estate, and your end-users enter the code on their device using the Partner Onboarding app.

Setting Up for Partner Code Onboarding

These are one-time steps you complete in HEM before distributing the code to your end users.

1. **Create an onboarding estate.** In Device Management, create an estate to act as your onboarding target (see Adding Estates). All devices that use your Partner Code will be moved into this estate.
2. **Create a permanent campaign.** In the Campaigns section, create a campaign containing the application(s) and configuration required for your all-in-one integration, and assign it to your onboarding estate (see Campaign Creation).

- Important: enable the Permanent Campaign switch to ensure the software is installed immediately whenever a device enters the estate.

3. **Share your estate ID with NMI.** Provide the ID of your onboarding estate to your Account Manager.

- Tip: you can find the estate ID in the URL when viewing your estate in HEM — it appears as the `currentEstate` parameter, for example:

```
https://<hem-address>/deviceManagement?currentEstate=12345678-0dc0-1234-a123-1234ac5a1e23
```

4. **Receive your Partner Code.** NMI will provision a Partner Code linked to your onboarding estate and return it to you. A single code maps to one estate and can be reused across all of your devices.
5. **Distribute the Partner Code** to your end users, along with their device.

The End-User Experience

A new device is first set up using the NMI Setup App, which guides the user through Wi-Fi and, optionally, Bluetooth setup, registers the device with the NMI backend, and downloads the NMI Transaction Initiator UI and Device Agent applications.

To complete partner onboarding, the user then opens the **Partner Onboarding app** and enters their Partner Code.

If the user enters a valid Partner Code, the device is moved into your onboarding estate and an update is triggered immediately, installing the applications from your permanent campaign. The user can exit the app without entering a code if they do not have one; the device then remains in its default estate and can be onboarded later.

If a code is entered incorrectly, the user is prompted to check it and try again. A code that is not recognised should be reported to you, so it can be confirmed against the one NMI provisioned.

The code is not case-sensitive and may be shown grouped with hyphens for readability (for example **000-019-VW**); the hyphens are optional and the code can be entered with or without them.

Role Access Levels

The following is a brief summary of the predefined roles and the permissions they provide to users.

PGP-ADMIN

Section	Access
Software Library	Read/Write/Update
Campaigns	Read/Write/Update
Device Management	Read/Write/Update
MDM Profile	Read/Write/Update
User Management	Read/Write/Update

PGP-CAMPAIGN-MANAGER

Section	Access
Software Library	Read/Write/Update
Campaigns	Read/Write/Update
Device Management	Read
MDM Profile	Read
User Management	None

PGP-DEVICE-MANAGER

Section	Access
Software Library	Read
Campaigns	Read
Device Management	Read/Write/Update
MDM Profile	Read
User Management	None

PGP-MDM-MANAGER

Section	Access
Software Library	Read
Campaigns	Read
Device Management	Read
MDM Profile	Read/Write/Update
User Management	None

PGP-VIEW-ONLY

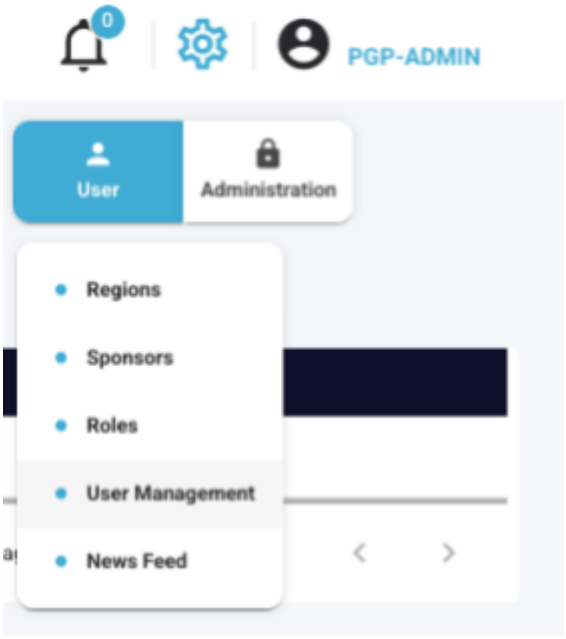
Section	Access
Software Library	Read
Campaigns	Read
Device Management	Read
MDM Profile	Read
User Management	None

Admin Controls

This section details admin-specific controls. Admin accounts also have the permissions granted to other roles. To avoid repetition please refer to the other role sections for further information on controls available to admins.

User Management

User management is accessible via the cog icon in the top right corner of HEM.



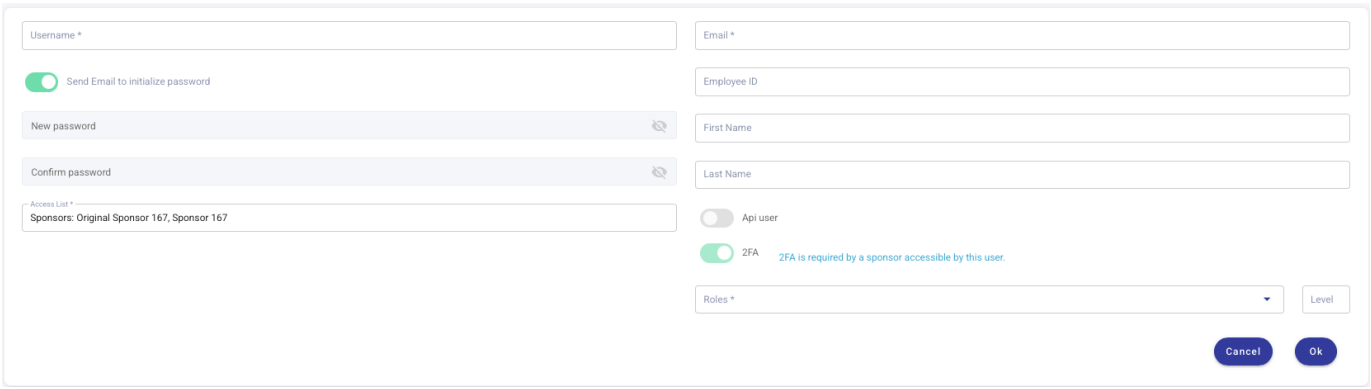
From the user management screen you'll be able to see details on all the users currently added to all sponsors your own account has access to.

Clicking the "..." next to the user will open a menu allowing you to perform the following actions:

- Edit user details
- Reset 2FA
- Set a new password
- Send a reset password email
- Deactivate the account
- Delete the account

CREATE A NEW ACCOUNT





To create a new user account click the + **New User** button. This will show the add user menu to input new user details and assign them their access type.

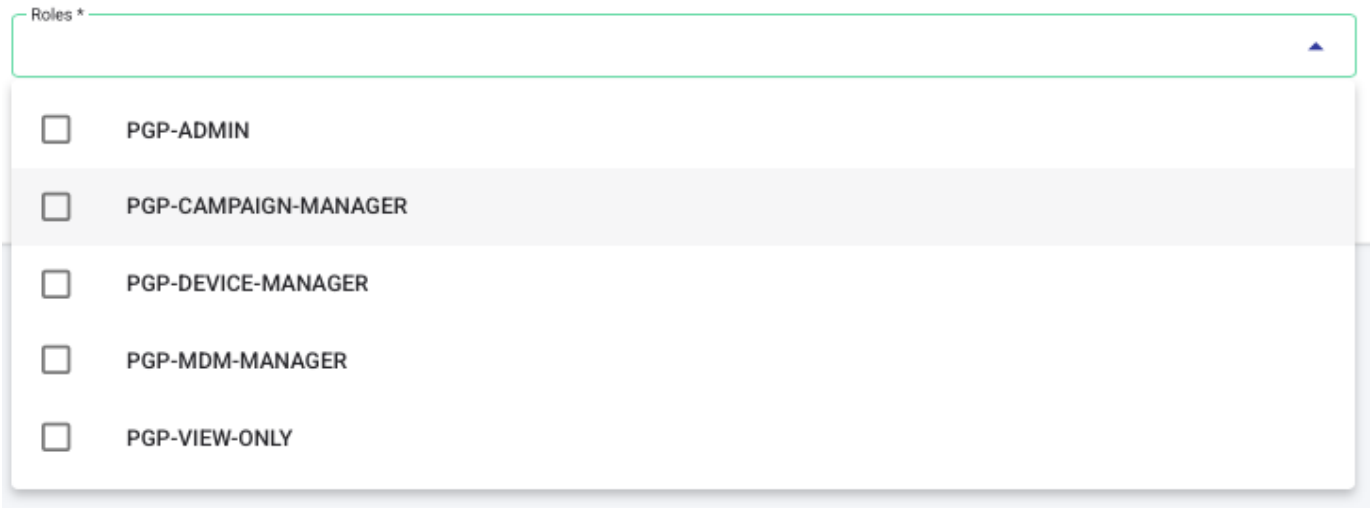


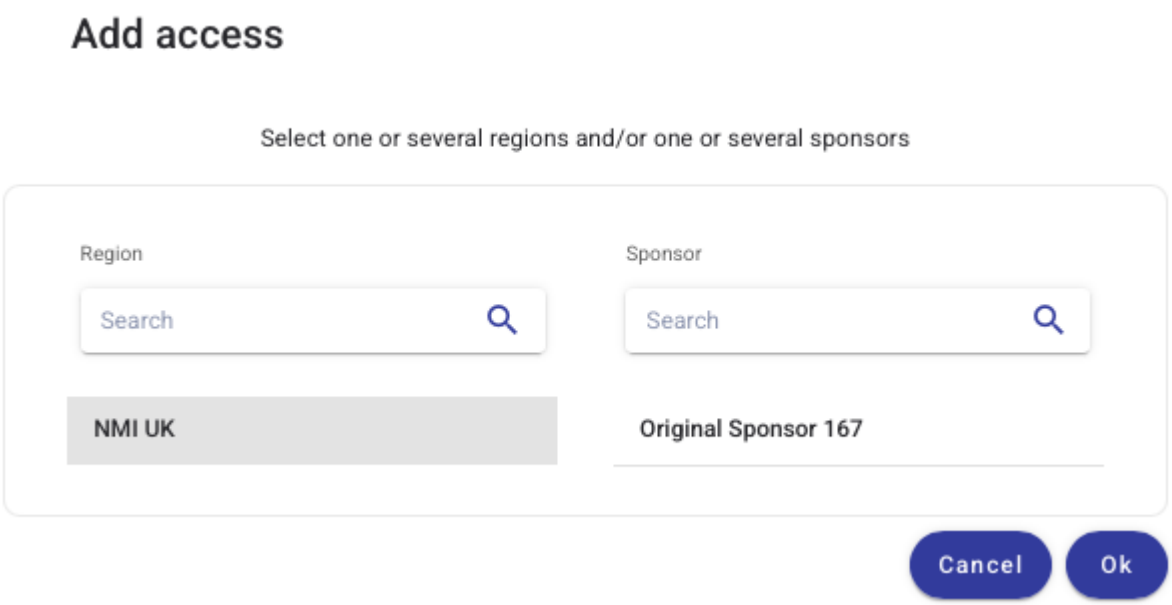
NMI does not impose any requirements around input and you only need to fill in fields that HEM requires or you would like to input. Required fields include:

- User Name
- Email — for the user being added
- Role — defines the level of access being granted

You can either set the new user's password, or have HEM send them an email to set their own password and 2FA. This is controlled by a toggle switch.

DEFINING ACCESS

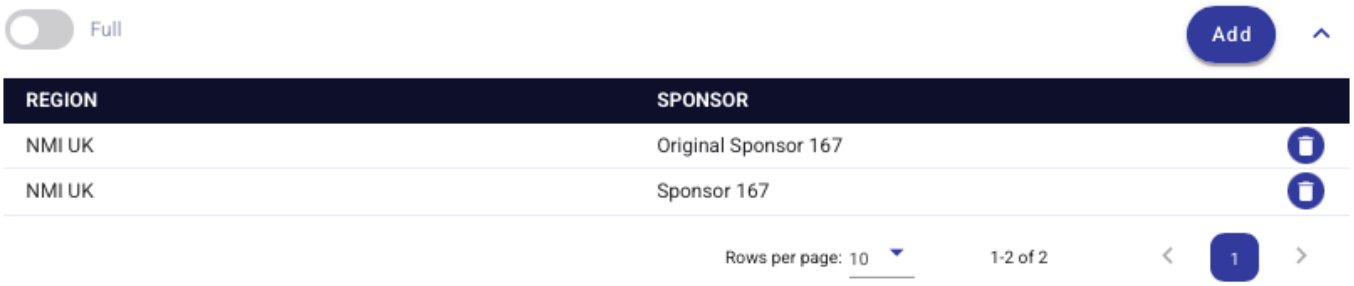




There are two parts to defining access:

- Sponsor Access
- User Role

By default new users will be assigned access to all sponsors that the admin creating the user has access to. To adjust this click on the "Access List" table to begin editing. You can then click the trash bin icon next to a sponsor to remove it, or click "Add" to add a sponsor.



When adding you will need to select the "NMI UK" Region, as all NMI partner sponsors will be under this region. You will then see a list of sponsors that your account has access to and can select ones that you wish to allow the new user access to.

To set the user's role click the roles drop-down box and select the roles you wish to give the new user. You can select as many as you want. The access granted by these roles can be reviewed in the Role Access Levels section above.

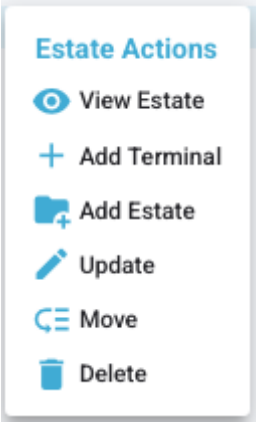
Once you're satisfied with the user's account information click the "Ok" button to complete the process of the user being added.

Device Manager Controls

The device manager role grants access to the Device Management page of HEM. This page allows you to manage your terminals and estates, building them into a logical structure to make management easier.

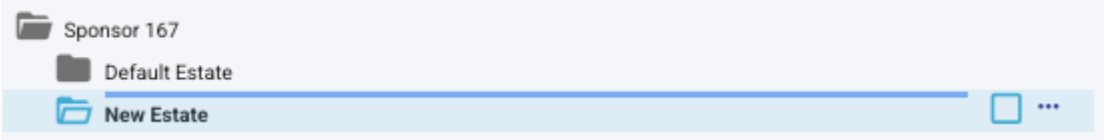
Managing Estates

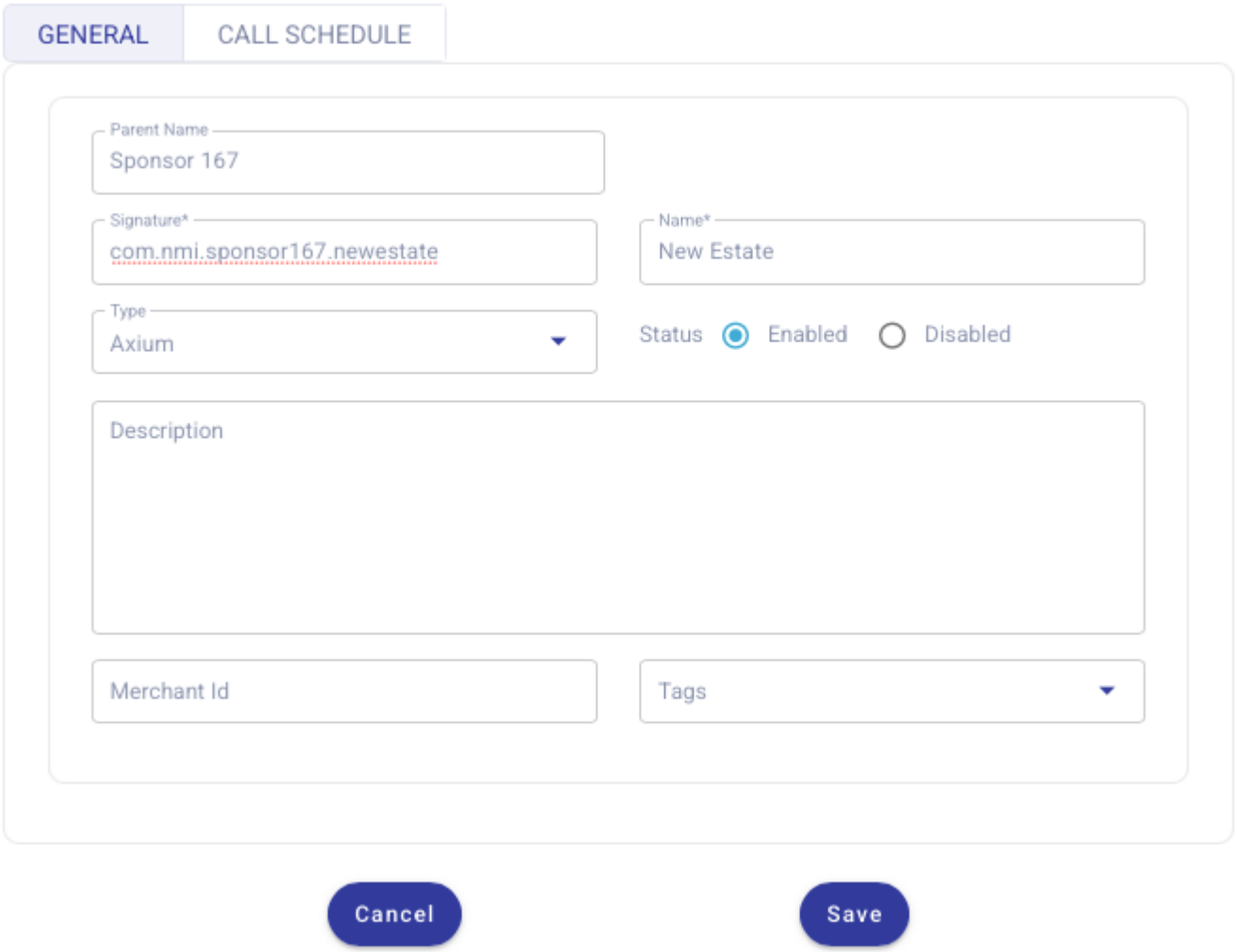
To add a terminal or estate within the Device Management section of HEM, select the parent estate you would like to add the terminal or estate to. This will reveal a "..." to the right of the estate name. Clicking this will reveal a menu showing the operations you can perform:



- **View Estate** — view the details of the currently selected estate
- **Add Terminal** — add a new terminal to the estate
- **Add Estate** — add a new sub-estate
- **Update** — update the details of the selected estate
- **Move** — move the selected estate/terminal to a new location
- **Delete** — delete the current estate/terminal

ADDING ESTATES





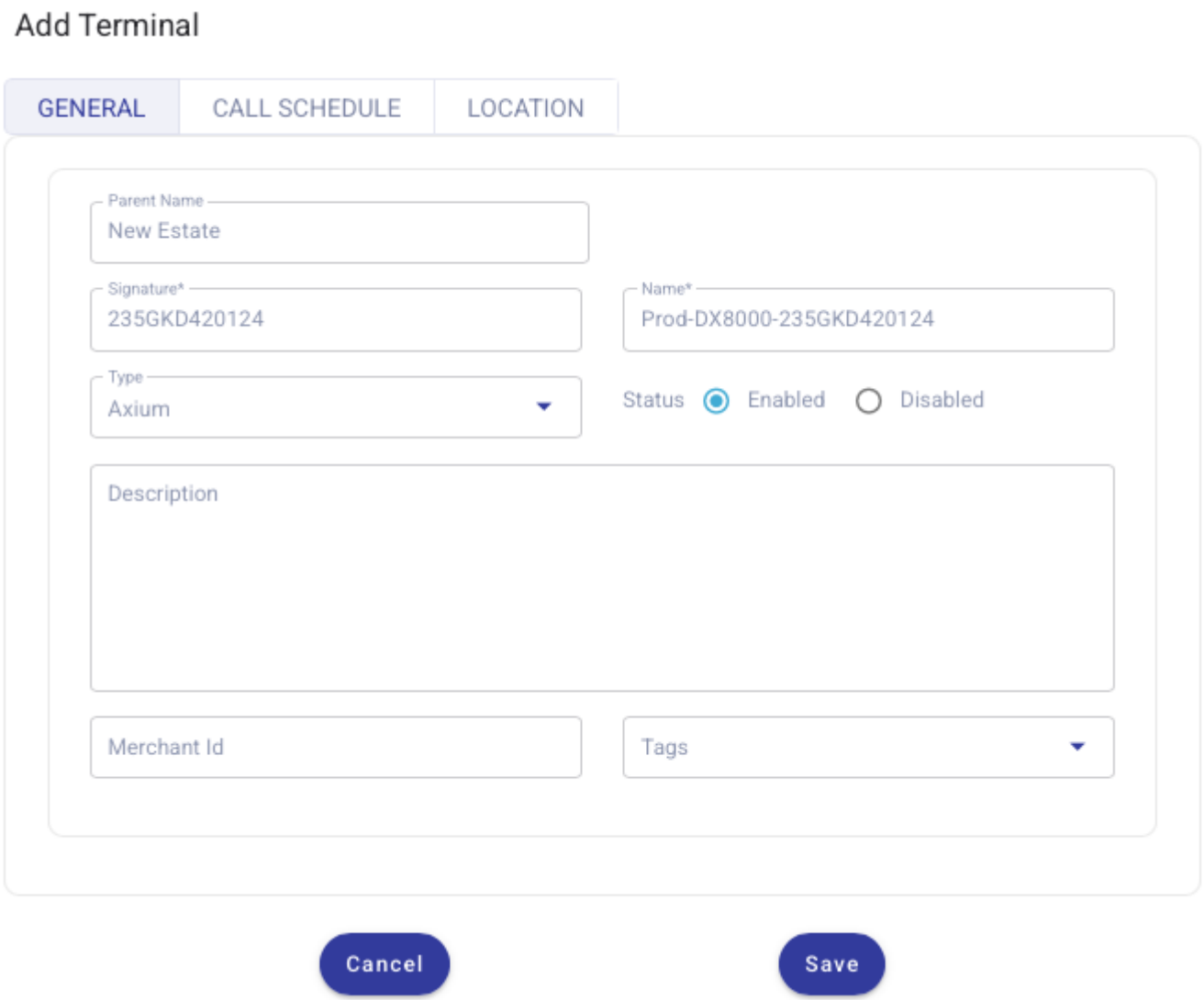
To add estates the following information must be filled out:

- **Estate Name** — what you would like the estate to be called
- **Signature** — a ***unique*** identifier for this estate. NMI recommends using a domain-based identifier.

The other fields can be filled out as you feel necessary; NMI doesn't place any restrictions on content.

When you are ready click the save button. The new estate can be found under the specified parent estate.

ADDING TERMINAL



If you have ordered your device through an NMI-approved distributor, you do not need to manually add terminals. New terminals will automatically appear in your default estate after their first registration with the NMI platform.

Attempting to add a device supplied by an NMI-approved distributor may fail, as the device is already registered within the NMI HEM region.

The following information is provided for completeness, in case manual terminal creation is required.

Adding a terminal is similar to creating an estate. You will be required to provide the following information:

- **Terminal Name** — the name assigned to the terminal
- **Signature** — the device's serial number

All other fields can be completed as needed. NMI does not impose restrictions on their content.

Call Schedule

The call schedule defines when a device calls home to HEM to see whether updates are required. The call schedule can be adjusted against the estate and synced to the terminals, or a terminal can have its own call schedule.

The fields are as follows:

- **Enable Call Scheduling** — enable or disable call scheduling. A device must call home in order for call scheduling updates to take effect.
- **Enable Load Balancing** — HEM will spread devices across the maintenance window with this enabled so they don't all call home at once.
- **Protect From Parent Update** — enabling this will mean a terminal won't receive updates to its call schedule from its parent estates.
- **Frequency days** — how often the device will call home, in days.
- **Retry Period** — if a HEM call fails, how often a device will try again within the call window.
- **Timezone Adjustment** — allows you to adjust the maintenance window to the terminal's local time.
- **Window** — the maintenance window that a device will call home in.
- **Reject calls out with Window** — HEM will reject calls that are not within the maintenance window.



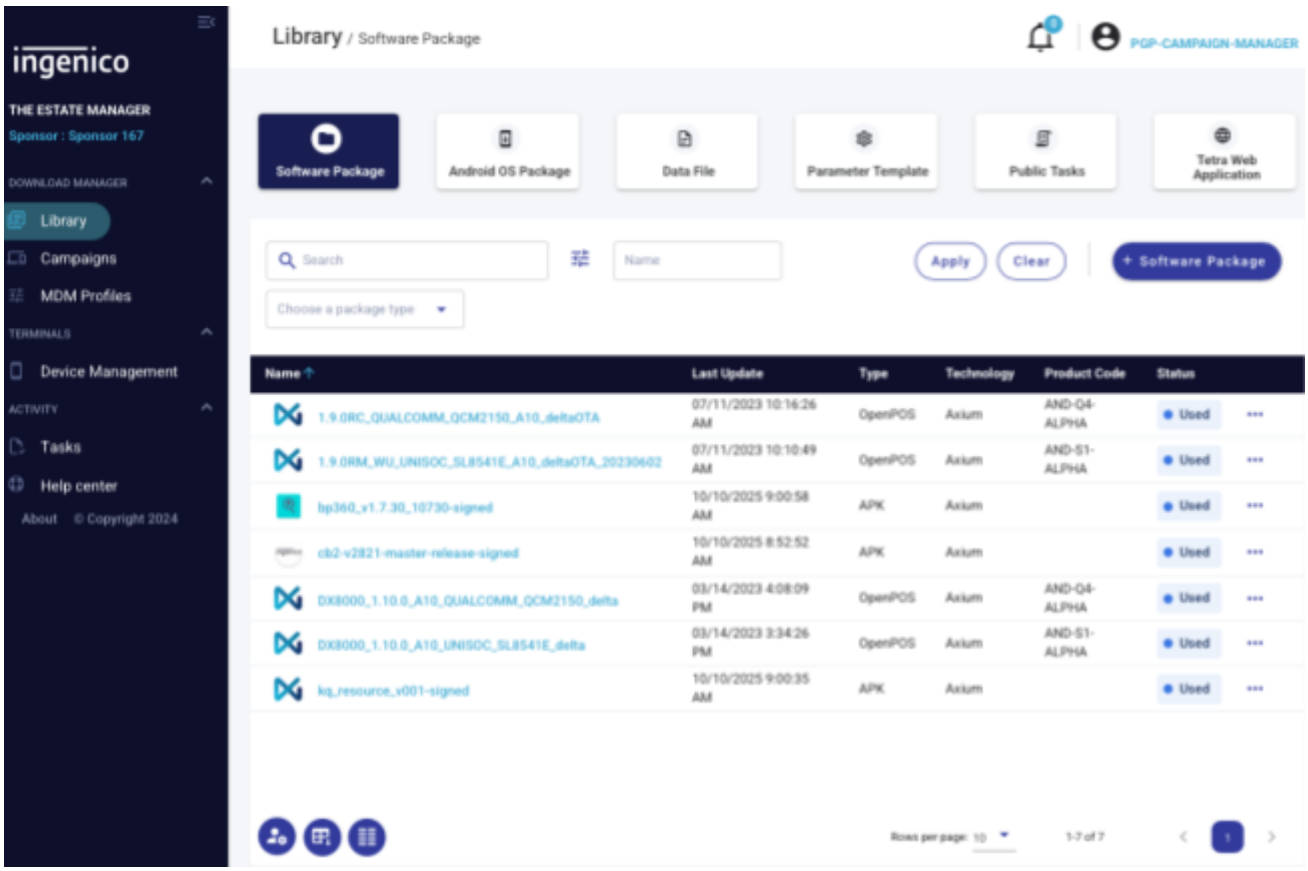
The call schedule is not pushed to the device when saved on HEM and is only updated on the terminal the next time the device calls home. If call scheduling is disabled on a device, a manual HEM update must be performed to re-enable it.

Setting the call schedule at the estate level will be applied to all devices within that estate, as long as those devices don't have their own call schedules applied.

Campaign Managers

The campaign manager role grants a user access to the Software Library and the Campaign section of HEM. In the Software Library you are able to upload your APKs to HEM; then in the campaign section you can choose how software will get applied to terminals.

Upload Software

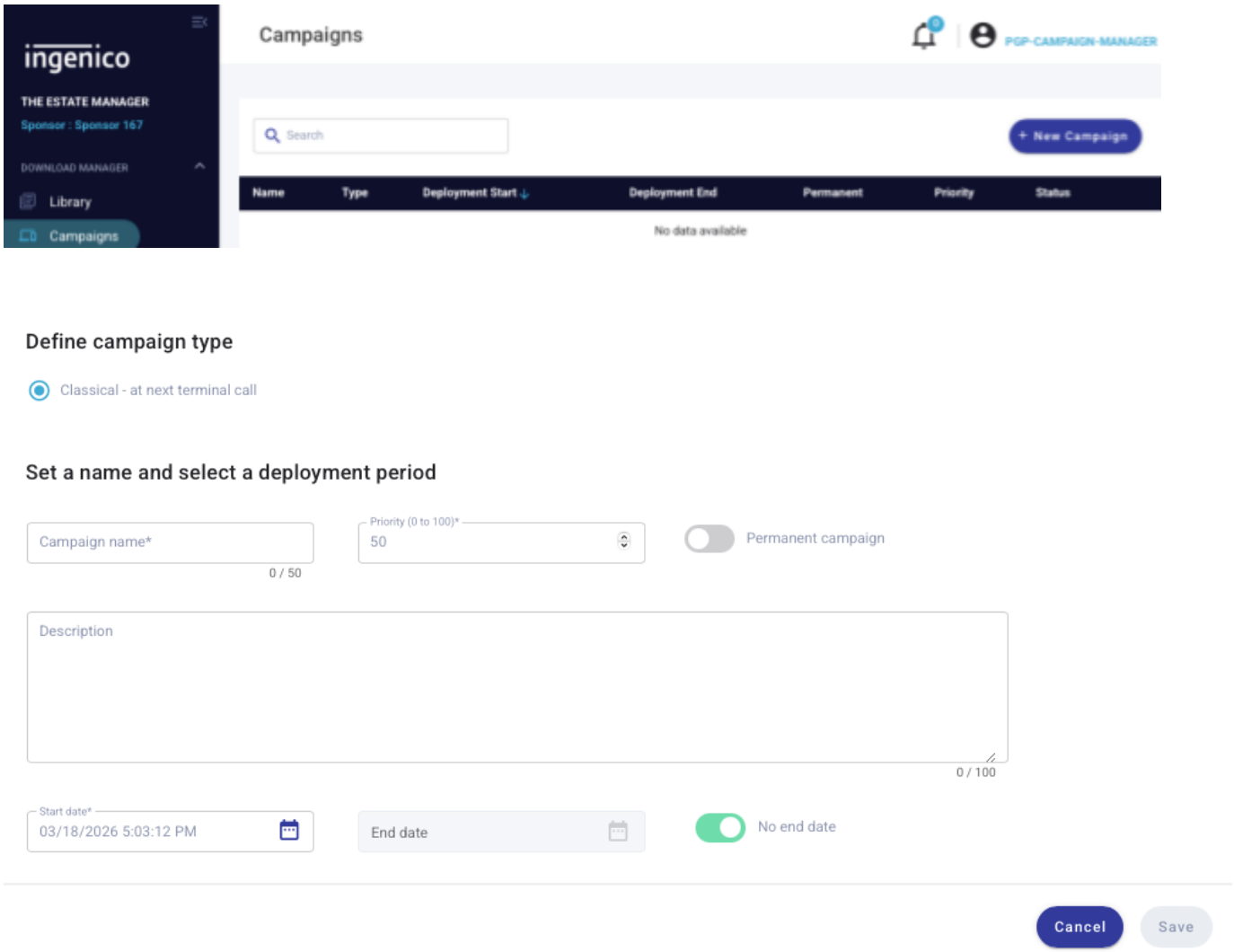


You can upload your APK to HEM from the library section of HEM. From the Software Package tab click on the "+ Software Package" button. This will bring up a file browser where you can select your APK for upload.

If you are intending to distribute your APK to production Axium devices it must be signed with a valid e-card from Ingenico's e-signing portal.

Campaign Creation

To create a campaign under the Campaigns section of HEM click the "+ New Campaign" button. This will bring up a window to allow you to input your campaign details.



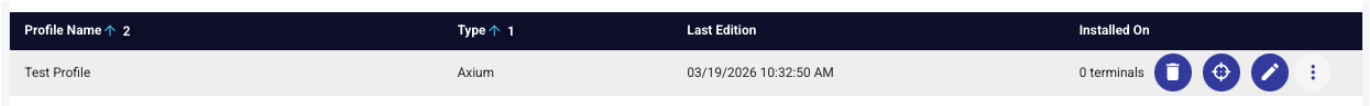
A campaign requires the following:

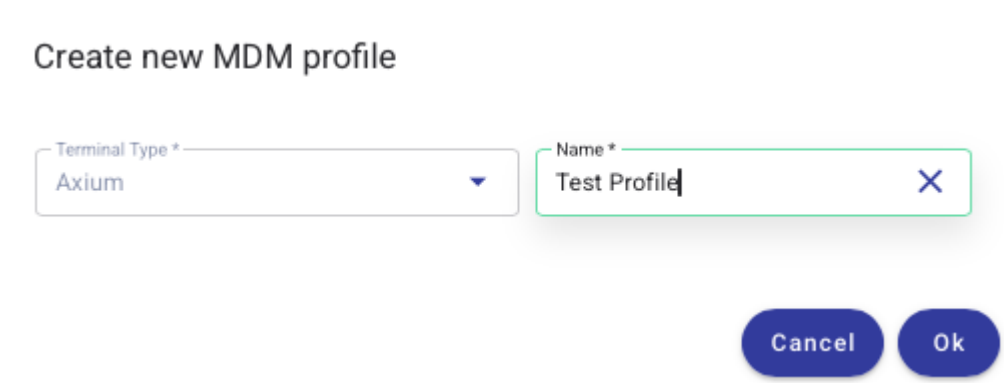
- **Campaign Name** — what you would like the campaign to be called.

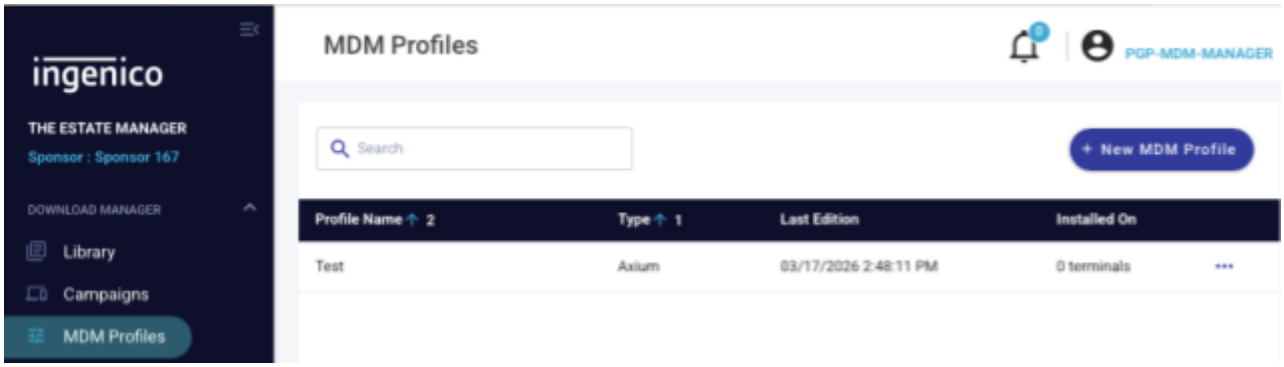
Useful to know:

- **Permanent Campaign Switch** — will mean the campaign is applied to devices even if they have already been applied previously. Note: this does not mean the same software is installed multiple times, just that it will be re-installed if removed.
- **Priority** — you can have multiple campaigns set against the same device or estate. These will be run in priority order, with 100 being the highest priority.

MDM Managers



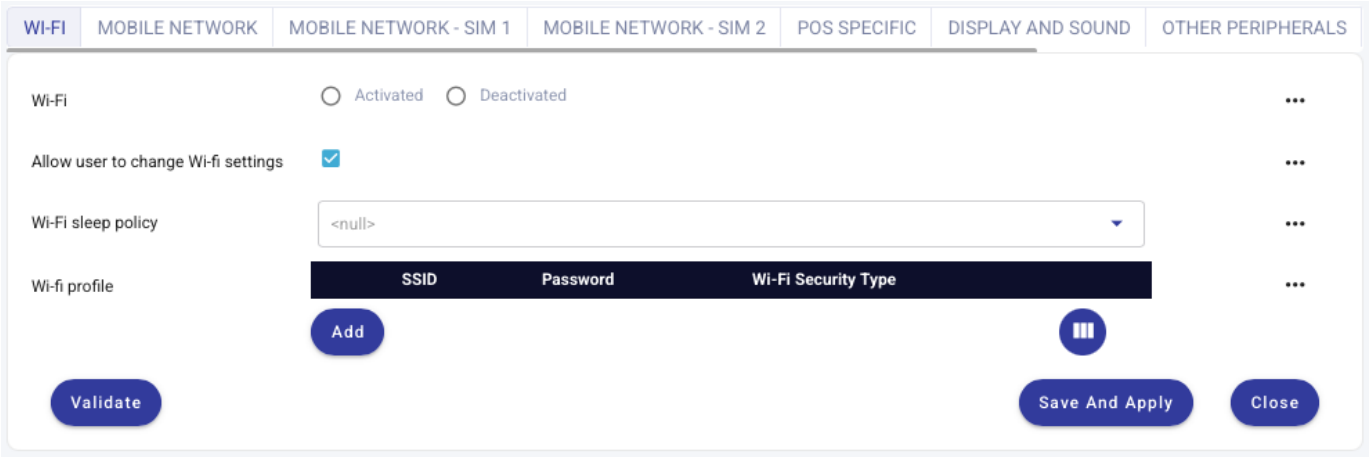




MDM Managers have access to the MDM Profiles section of HEM. Here they are able to create, modify and apply MDM profiles to devices.

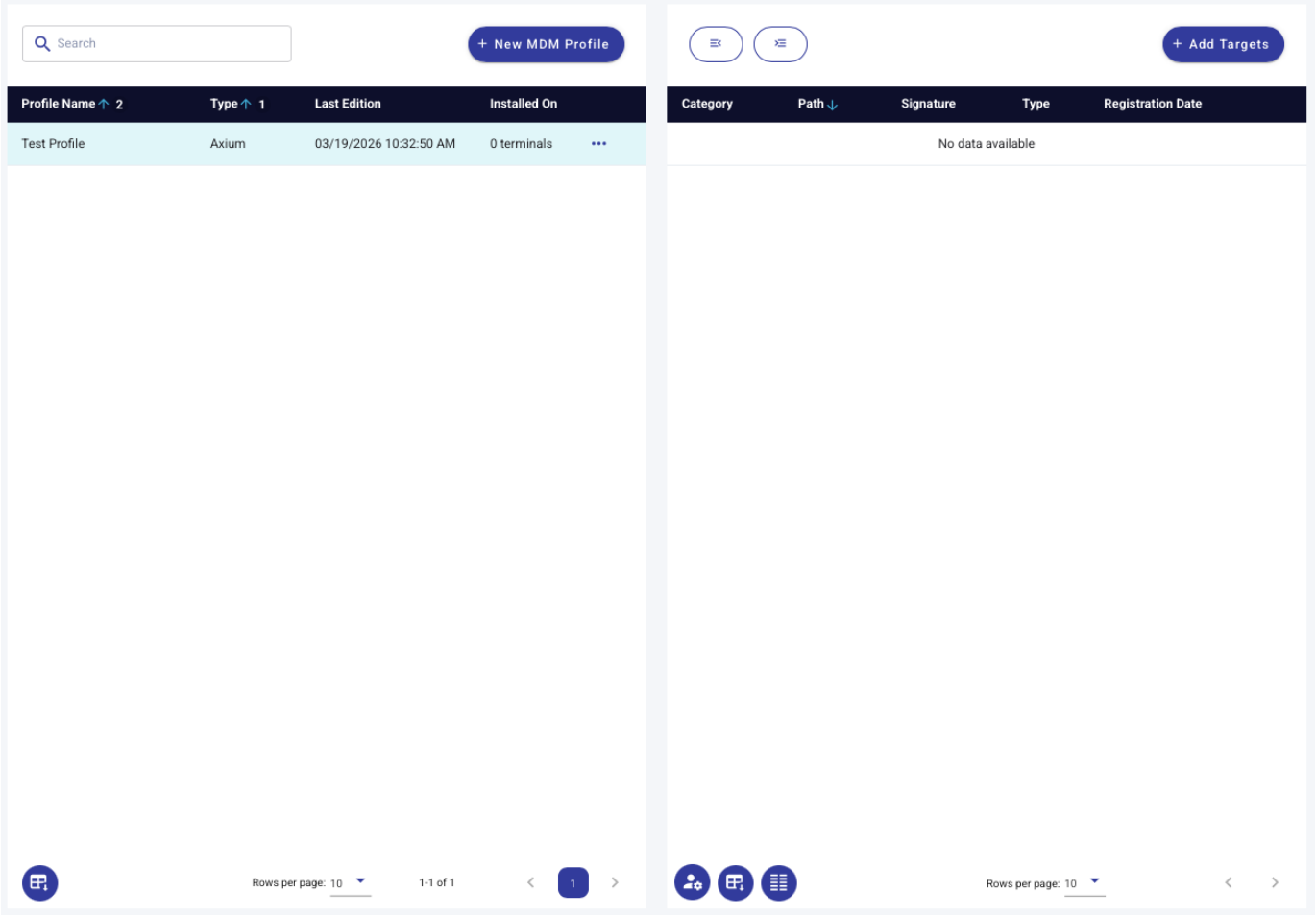
MDM profiles are applied in a similar way to campaigns. First you need to create one using the "+ New MDM Profile" button.

Set the type of profile to "Axium" and use a name that makes sense for your needs. Then click "Ok" and HEM will allow you to fill out details for your MDM profile.

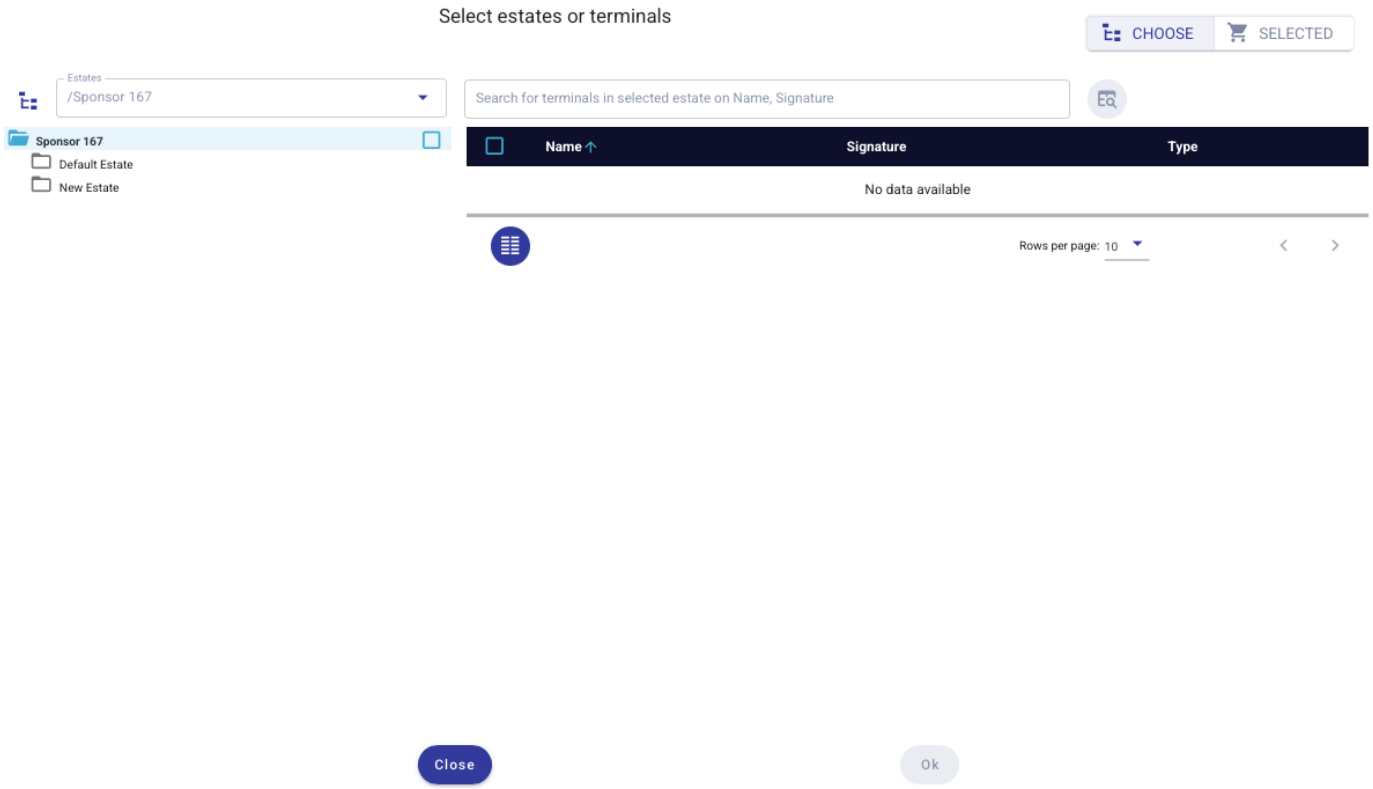


Once complete you can validate the profile using the "Validate" button and then "Save and Apply". You'll notice your profile is then available to be applied to devices. You will do this using the "..." available at the right of the profile.

This allows you to delete, apply or edit your profile. Clicking the cross-hairs symbol will show you the current devices/estates that are currently being targeted.



More targets can be added using the "+ Add Targets" button. This will allow you to explore your current estates and select any you wish the MDM profile to apply to.



Targets will receive the MDM profile next time they call home.

Viewers

NMI has provided the `PGP-VIEW-ONLY` role which allows a user "view" access to your HEM estate. They will be able to view all pages **except** the User Management page.

5. Application Signing

Any application intended to run on a production Axium device must be signed using a valid e-signing card provided by Ingenico.

To sign an application, you will need access to Ingenico's e-signing portal and a provisioned e-signing card. NMI will assist in enabling this via your Account Manager when access to HEM is requested.

Signing

Signing is a two-step process involving two user roles:

- Pusher
- Validator

The Pusher uploads the application (APK) to the Ingenico e-signing portal and submits it for validation. The Validator reviews the APK to ensure it is suitable for signing and, if approved, completes the signing process.

NMI will provision four accounts for the Ingenico e-signing portal:

- Two Pushers
- Two Validators

These accounts are included in the cost of enrollment. Additional accounts can be provisioned via your Account Manager for an additional fee.