



Locking Down Your Enterprise Device

DevTalk – 18th October 2017





Updated from a blog written in April 2017: <https://developer.zebra.com/community/android/android-forums/android-blogs/blog/2017/04/10/locking-down-your-device>

Darryn Campbell
Senior Software Architect

Agenda:

- Overview of options
- Android Enterprise device with Device Owner
 - AKA COSU devices, managed devices & single purpose devices
- Mobility eXtensions (MX)
- Enterprise Home Screen (EHS)
- Enterprise Browser (EB)
- Summary & Resources

Options for locking down your device

Option	Description
	Managed Android devices. COSU device use cases, <u>test</u> on Marshmallow or above. End to end support from Zebra coming in future releases, watch out for announcements. <i>NOT proprietary to Zebra</i>
	Mobility eXtensions (MX) Only available on Zebra devices. Full set of capabilities to restrict what a user can do on the device and <u>secure your deployment</u> . Supported by partner EMMs (MDMs). Accessible at runtime or during staging.
	Enterprise Home Screen (EHS) Post installable tool that runs on Zebra devices only. Replaces the default Android launcher with one designed for Enterprise. Offers customized branding and a <i>selection of lock down functionality</i> .
	Enterprise Browser (EB) Post installable application that runs on Zebra devices only. Run an HTML, CSS & JavaScript application leveraging the full range of device hardware. Offers <i>configuration options to lock the user to your app</i> .



Managed Android Devices

- NOT proprietary to Zebra
- Support coming in future releases but the principles are testable today in Marshmallow (just not the end to end soln)
- Aligns with Google's continued focus on Enterprise use cases
- Natural progression of the 'Android For Work' feature set first announced in Lollipop back in 2014.
 - The 'Android for work' name is now retired but the feature set is definitely still supported, now considered part of standard Android.
- Google's focus on a wide range of use cases: BYOD, COPE, COSU and more coming on Oreo
 - Zebra devices are COSU or 'Corporate Owned, Single Use'.

First, Provision devices

In order for your device to be managed there needs to be an application on the device which will act as a device owner (DO), hence why this is also called DO mode. This application, called a Device Policy Controller (DPC) has access to the DevicePolicyManager APIs

(<https://developer.android.com/reference/android/app/admin/DevicePolicyManager.html>) and will often be provided by your chosen EMM.

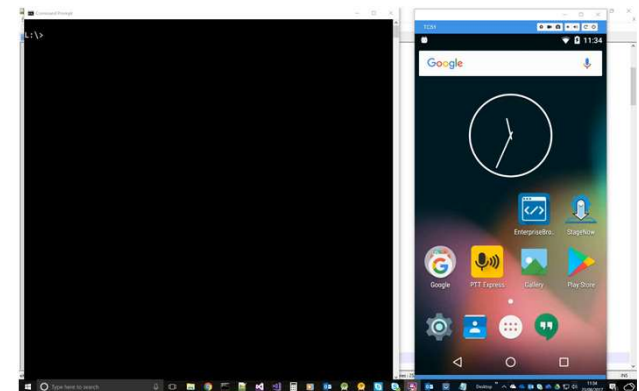
So, out of the box we need to tell Android to install a DPC and set it as the DO. There are a number of ways to do this:

- Via NFC, bump against a pre-configured tag or device, as detailed here: <https://source.android.com/devices/tech/admin/provision>
- By scanning a barcode. See Google IO 2016 demo (<https://www.youtube.com/watch?v=Za0OQo8DRM4&feature=youtu.be&t=874>). Zebra devices will also support the hardware scanner.
- By setting an adb command, most useful for testing. Demonstrated on the next slide
- Look out for additional options when this mode is fully supported by Zebra

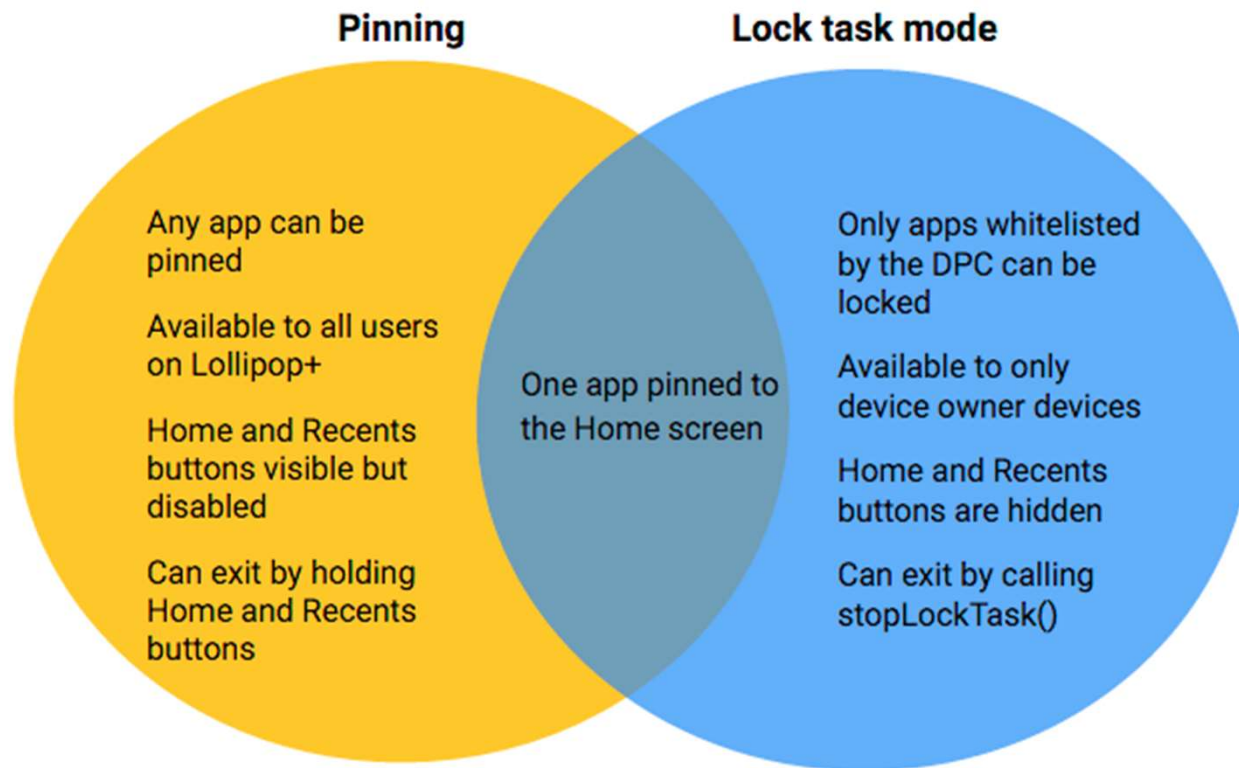
To **test** on a TC51 Marshmallow device we can put it into DO mode:

- Our DPC will be Google's TestDPC application, <https://github.com/googlesamples/android-testdpc>. Also available from the PlayStore
- Remove all previous accounts from the device
- Set TestDPC as the DO via ADB:
- `adb shell dpm set-device-owner "com.afwsamples.testdpc/.DeviceAdminReceiver"`

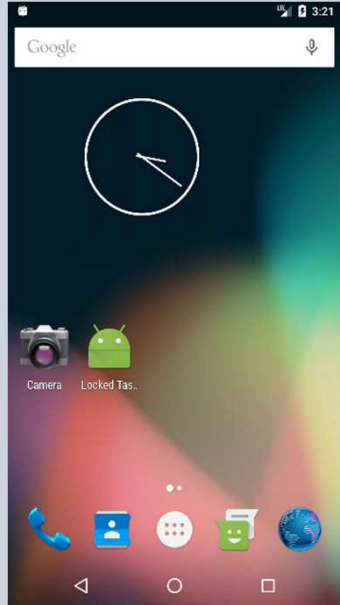
Demonstration:



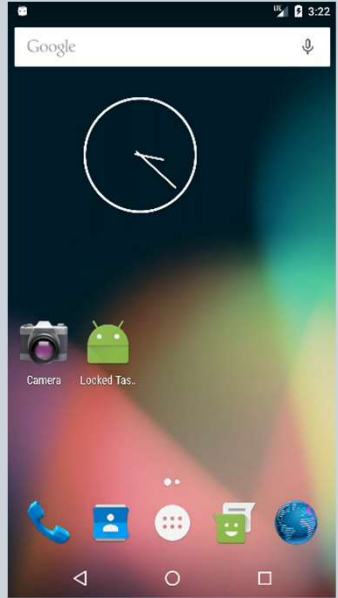
Lock Task Mode vs. Screen Pinning



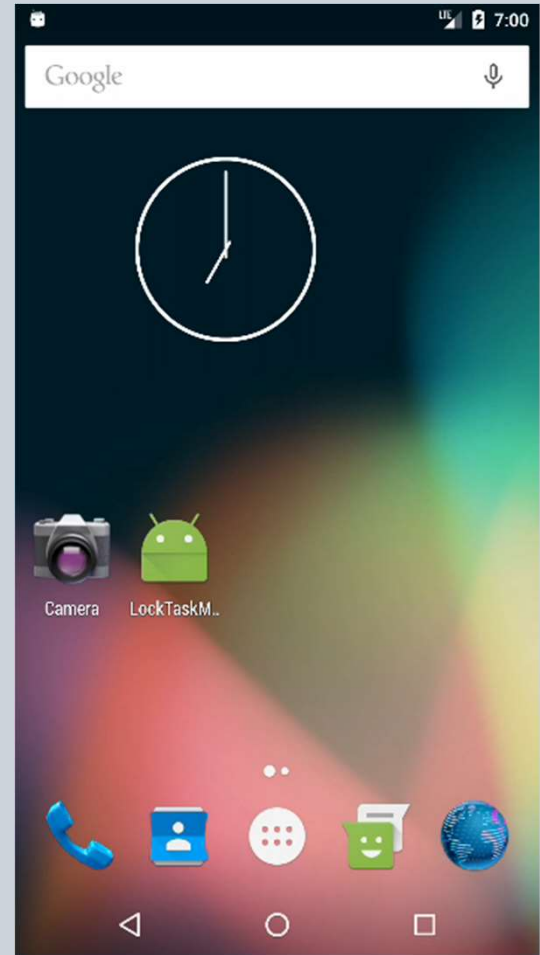
Base Case: Screen Pinning

Description	Demonstration
<p>Even running on an Android Marshmallow device, calling <code>startLockTask()</code> does not automatically enter lock task mode.</p> <p>The application must first be added to the lock task list, controlled by the DPC to whitelist the application.</p> <p>If an application is not whitelisted then calling <code>startLockTask()</code> will just result in application pinning.</p>	 <p>The screenshot shows an Android phone screen with a dark blue background. At the top is a Google search bar. Below it is a large white clock face. Further down are two app icons: a camera icon labeled 'Camera' and a green Android robot icon labeled 'Locked Tas...'. At the bottom is a dock with five icons: a blue phone icon, a blue person icon, a white circle with three dots, a green speech bubble icon, and a blue globe icon. The status bar at the very top shows the time as 3:21 and various system icons.</p>

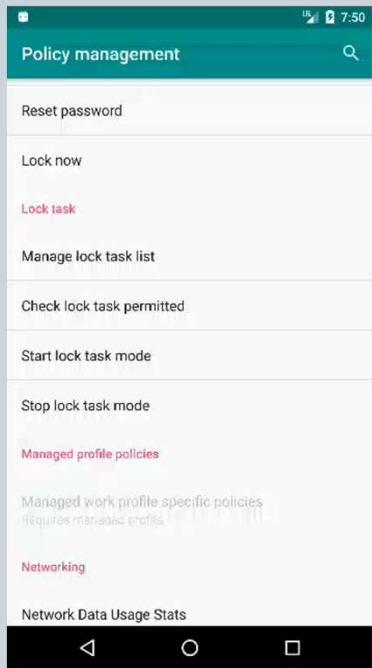
Invoking lock task mode from an application

Description	Demonstration
<p>If an application is whitelisted by the DPC, calling the <code>startLockTask()</code> API will cause the device to enter lock task mode.</p> <p>From this mode, the user cannot leave the displayed application.</p> <p>Note: the 'Screen pinned' toasts do not come from the test application, the Android framework is providing misleading information</p>	

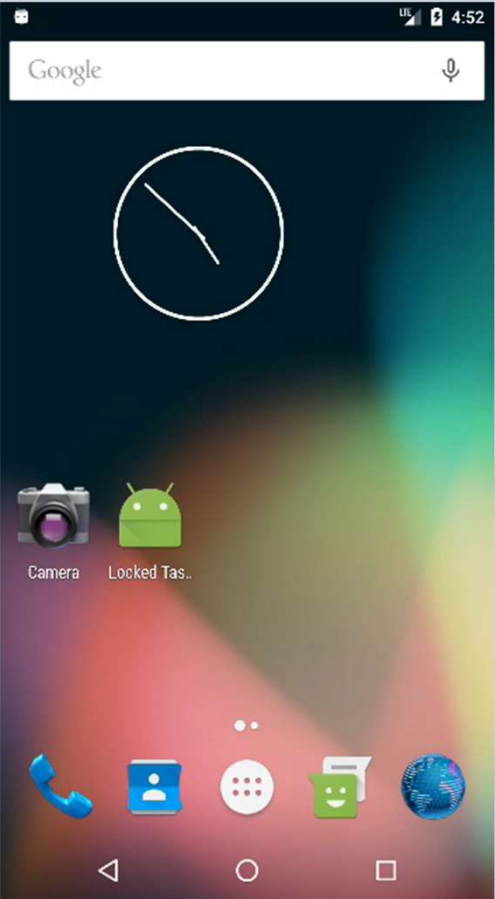
Entering lock task from the DPC

Description	Demonstration
<p>Calling <code>startLockTask()</code> from the DPC will cause the device to enter lock task mode. From this mode, you can also enter additional whitelisted applications by sending an intent (for the video, this is done via adb, in a real app you would call <code>startActivity()</code>) resulting in a stack of whitelisted applications.</p> <p>Pressing back will cause the stack to unwind and the original (DPC) application to show.</p> <p>Finally, calling <code>stopLockTask()</code> from an application that did not initiate lock task mode will result in a security error; if you call start / stop lockTask in that order from the launched app the stack will again unwind and the DPC app be shown.</p>	

Checking the state of lock task mode

Description	Demonstration
<p>There are a couple of APIs which can be used to determine lock task mode.</p> <p>Firstly the DevicePolicyManager has an API to determine whether your application is whitelisted but that must be called from the DPC.</p> <p>The activity also has access to getLockTaskModeState() which returns whether or not the application is currently locked or pinned</p>	 <p>The screenshot shows the 'Policy management' screen on an Android device. The screen has a teal header with the title 'Policy management' and a search icon. Below the header, there is a list of options: 'Reset password', 'Lock now', 'Lock task' (highlighted in red), 'Manage lock task list', 'Check lock task permitted', 'Start lock task mode', 'Stop lock task mode', 'Managed profile policies' (highlighted in red), 'Managed work profile specific policies' (with a subtext 'requires managed profile'), 'Networking' (highlighted in red), and 'Network Data Usage Stats'. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps buttons.</p>

The lockTaskMode attribute in the Application Manifest

Description	Demonstration
<p>Although the documentation for lockTaskMode will list 'normal', 'never', 'always' and 'if_whitelisted'; note that 'never' and 'always' are only available to system apps and privileged apps so a typical end user application will only be able to choose between 'normal' and 'if_whitelisted'.</p> <p>The examples listed so far have shown the effect of 'normal' which is also the default behaviour. If you specify 'if_whitelisted' then the application will launch straight into lock task mode.</p> <p>The video demonstrates an application configured with 'if whitelisted' being launched twice, the first time is it not white-listed and the second time it is white-listed.</p>	



Mobility eXtensions (MX)

- Proprietary to Zebra
- Enterprise focused functionality on top of standard Android
- Supports all Zebra devices from JellyBean upwards.
 - As functionality was added to MX, the corresponding functionality needs to be present on the device.
 - Older devices may not support newer MX features
 - MX is updated on device through a BSP update
- **Best technique for securing your device apart from DO mode.**
 - Can (& should) be used to supplement other techniques e.g. EHS or EB
- Will continue to be improved
- Can be accessed at runtime (EMDK) or during provisioning (StageNow or supported EMMs)



MX Features to lock down your device:

Profile	Features
Access Manager	Application whitelisting Prevent installation & running of non-whitelisted apps Secure which apps can run through use of app signatures Prevent ability of non-whitelisted apps to invoke other profiles Prevent access to system settings
UI Manager	Prevent access to the notification shade and settings therein Prevent access to the tiles which enable / disable wireless etc. Prevent Clipboard access Disable home key
Application Manager	Disable applications from running Prevent access to application <i>info</i> settings via the Settings UI
Settings Manager	Prevent access to application settings via the Settings UI

New features are being continually added, in the pipeline today are more ways to lock down the device.



Mobility eXtensions (MX) at Runtime

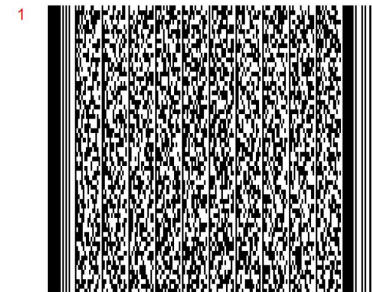
- This sounds familiar?
 - I demo'd this in the February 2017 DevTalk:
<https://developer.zebra.com/community/developer-events/blog/2017/01/27/dev-talk-feb-22-best-enterprise-features-in-android-m-n>
 - Sample app which uses MX at runtime is here:
<https://github.com/darryncampbell/MxLockDown>
 - Demo video showing the sample app in action is here:
<https://youtu.be/C9OCWsOt4W4>
 - Also demo'd this at the NALA and EMEA AppForums if you were there but unfortunately the live demos were not recorded.
- Boo! Give us something new!



Mobility eXtensions (MX) during Provisioning

Created a StageNow barcode for MX6.0 which does the following:

- **UI Manager:** Disables the notification shade and home key
- **Access manager:** Applies 'Single User with Whitelist mode'; reduces the available settings; adds my test app to the whitelist
 - adds com.symbol.emdkservice as detailed in the docs but that is not required here.
- **Key Mapping Manager:** Change the Volume up and down keys to remap them to an unused key (F12).
 - Suppressing the 'back' and 'recent' keys is not yet supported on my device so a kiosk option would need to be sought in production.
- **Application Manager:** Prevent Chrome from running, disable access to app info for all applications





Mobility eXtensions (MX)



```
<?xml version="1.0" encoding="utf-8"?>
<wap-provisioningdoc>
  <!--SettingID=7-->
  <characteristic version="5.2" type="UiMgr">
    <parm name="ClipboardUsage" value="2" />
    <parm name="HomeKeyUsage" value="2" />
    <parm name="NotificationPullDown" value="2" />
  </characteristic>
  <!--SettingID=9-->
  <characteristic version="4.3" type="AccessMgr">
    <parm name="OperationMode" value="2" />
    <parm name="SystemSettings" value="2" />
    <parm name="DeletePackagesAction" value="2" />
    <parm name="AddPackagesAction" value="1" />
    <parm name="AddPackageNames"
value="com.darryncampbell.presentations.mxlockdown,com.symbol.emdkservice" />
    <parm name="AddPackagesActionAllowXML" value="1" />
    <parm name="AddPackageNamesAllowXML"
value="com.darryncampbell.presentations.mxlockdown,com.symbol.emdkservice" />
    <parm name="AllowSubmitXMLAction" value="2" />
  </characteristic>
  <!--SettingID=13-->
  <characteristic version="4.4" type="KeyMappingMgr">
    <parm name="Action" value="1" />
    <characteristic type="KeyMapping">
      <parm name="KeyIdentifier" value="VOLUMEUP" />
      <characteristic type="BaseTable">
        <parm name="BaseBehavior" value="2" />
        <parm name="BaseKeyCode" value="142" />
      </characteristic>
    </characteristic>
  </characteristic>
  <!--SettingID=19-->
  <characteristic version="4.4" type="KeyMappingMgr">
    <parm name="Action" value="1" />
    <characteristic type="KeyMapping">
      <parm name="KeyIdentifier" value="VOLUMEDOWN" />
      <characteristic type="BaseTable">
        <parm name="BaseBehavior" value="2" />
        <parm name="BaseKeyCode" value="142" />
      </characteristic>
    </characteristic>
  </characteristic>
  <!--SettingID=14-->
  <characteristic version="5.1" type="AppMgr">
    <parm name="Action" value="DisableApplication" />
    <parm name="Package" value="com.android.chrome" />
    <parm name="AccessAppInfoAction" value="DisableAccessAllInfo" />
  </characteristic>
</wap-provisioningdoc>
```

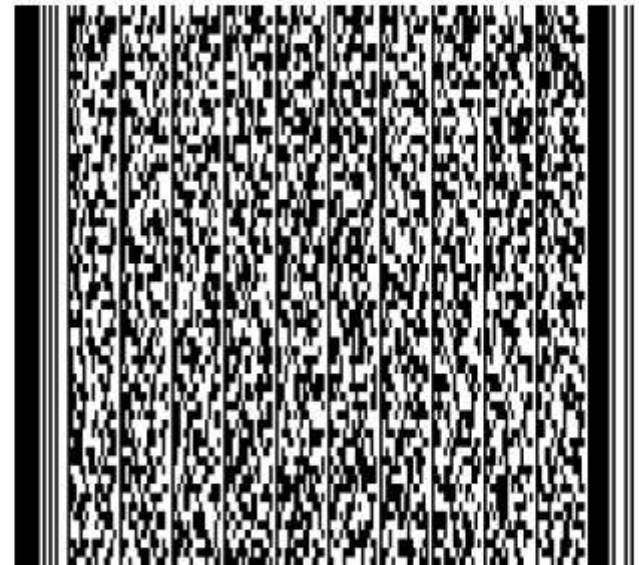
Profile Name: LockDownDevice

Barcode Type: PDF417

Minimum Compatible MX Version: 6.0

Scan Barcodes with StageNow Client:

1



2





```
<?xml version="1.0" encoding="utf-8"?>
<wap-provisioningdoc>
  <!--SettingID=8-->
  <characteristic version="5.2" type="UiMgr">
    <parm name="HomeKeyUsage" value="1" />
    <parm name="NotificationPullDown" value="1" />
  </characteristic>
  <!--SettingID=11-->
  <characteristic version="4.3" type="AccessMgr">
    <parm name="OperationMode" value="2" />
    <parm name="SystemSettings" value="1" />
  </characteristic>
  <!--SettingID=10-->
  <characteristic version="4.3" type="AccessMgr">
    <parm name="OperationMode" value="1" />
  </characteristic>
  <!--SettingID=15-->
  <characteristic version="4.4" type="KeyMappingMgr">
    <parm name="Action" value="2" />
  </characteristic>
  <!--SettingID=16-->
  <characteristic version="5.1" type="AppMgr">
    <parm name="Action" value="EnableApplication" />
    <parm name="Package" value="com.android.chrome" />
    <parm name="AccessAppInfoAction" value="EnableAccessAllInfo" />
  </characteristic>
</wap-provisioningdoc>
```



StageNow

Profile Name: UndoLockDownDevice

Barcode Type: PDF417

Minimum Compatible MX Version: 6.0

Scan Barcodes with StageNow Client:

1

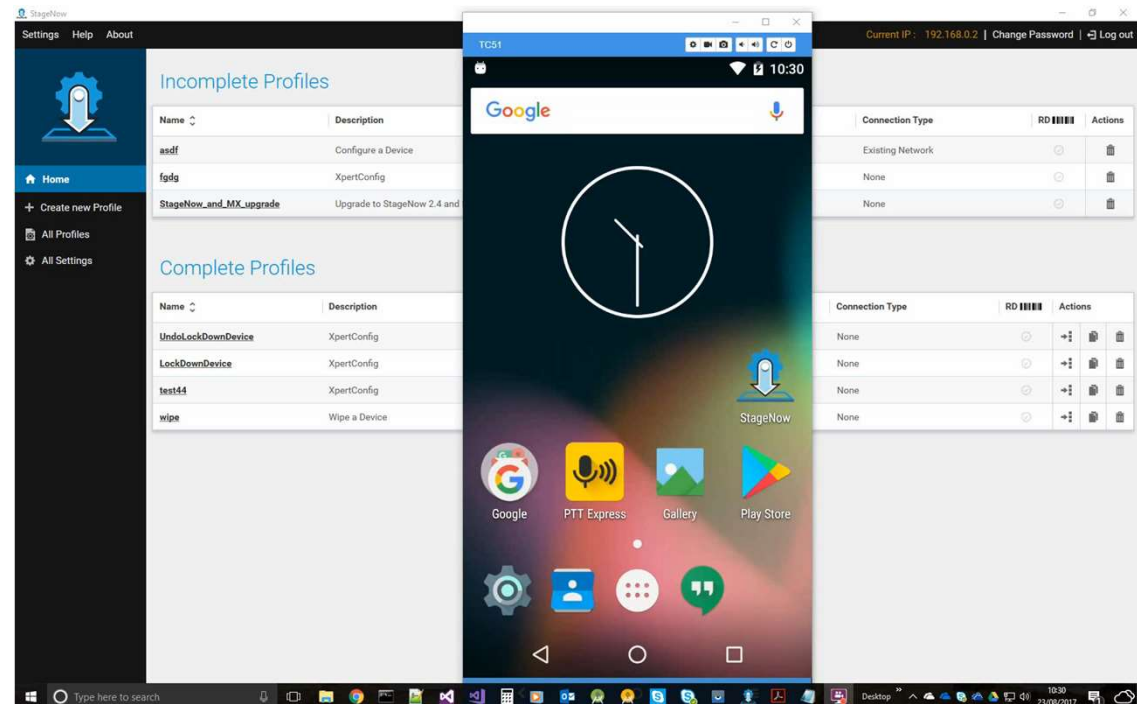




Mobility eXtensions (MX) during Provisioning

Demo:

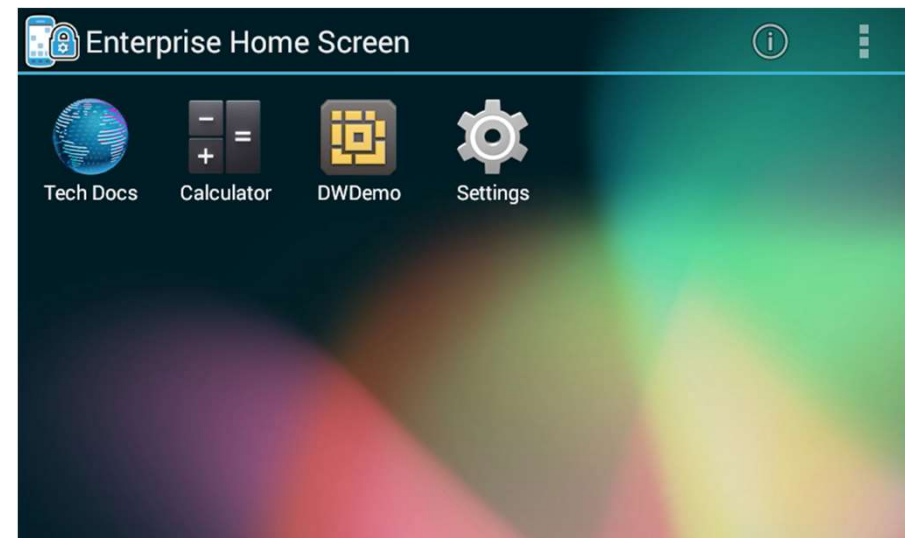
- Unauthorized user app
- Home key
- Notification shade and quick settings
- Can disable apps via app info
- Full settings available
- Can launch Chrome
- Can adjust volume





Enterprise Home Screen – What is it?

- Launcher application designed primarily for Enterprise
- Enables company branding / iconography
- **Should be used in conjunction with MX for a complete solution**
- Exposes a subset of security features not (yet) available through MX
 - E.g. disable USB debugging
 - Notably for this presentation, **Kiosk Mode**
- **Demonstration:**





Enterprise Home Screen – Kiosk Mode Demo

```
<kiosk>
```

```
<application label="Calculator"
```

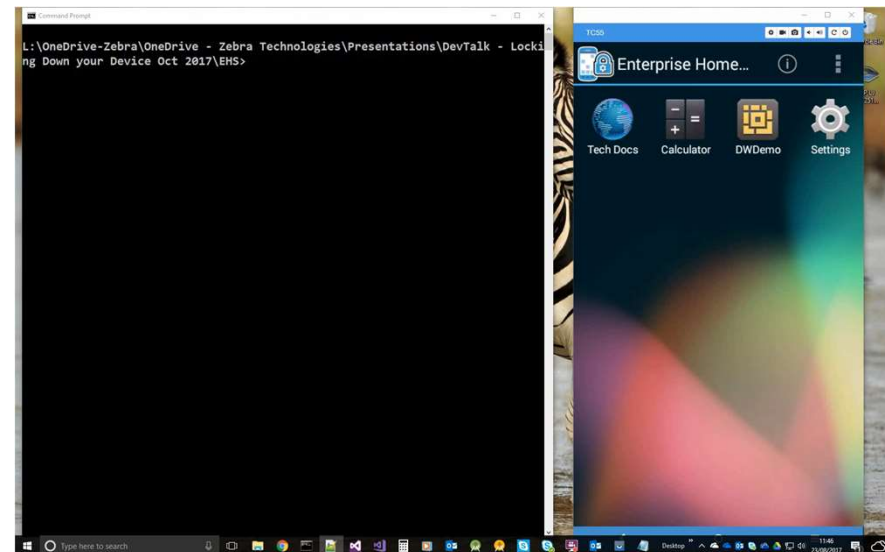
```
package="com.android.calculator2" activity=""/>
```

```
</kiosk>
```

```
<kiosk_mode_enabled> 1 </kiosk_mode_enabled>
```

To enable or disable push an appropriate config to the device:

```
adb push ehs_kiosk_calc.xml /enterprise/usr/enterprisehomescreen.xml
```





Enterprise Browser – New “Custom Kiosk Mode” features

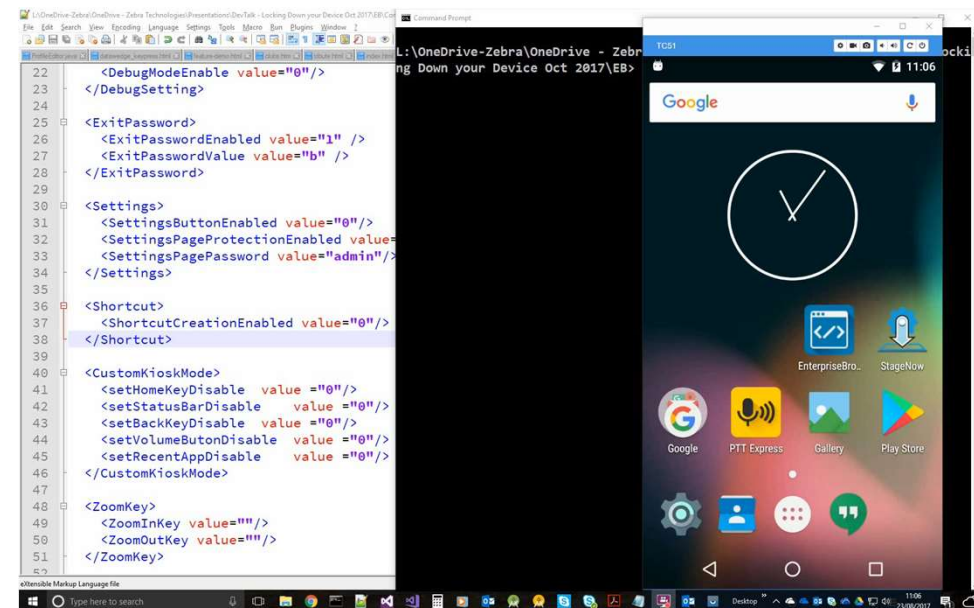
- Available in **EB 1.7 or higher**, released July 2017
- Official documentation is here: <http://techdocs.zebra.com/enterprise-browser/1-7/guide/configreference/>
- Similar to EHS, EB kiosk mode should be used in conjunction with MX for a complete solution.

Setting	Description
SetHomeKeyDisable	Disables the device home key in the navigation bar
SetStatusBarDisable	Disables the notification pull-down area. The status bar itself is still visible so you can still see the battery / signal levels.
SetBackKeyDisable	Disables the device back key in the navigation bar
SetVolumeButonDisable	Disables both the volume up and down keys
SetRecentAppDisable	Disables the device recent apps key in the navigation bar







Enterprise Browser – Demo of Custom Kiosk Mode

- Below video demonstrates pushing two config files to EB
 - First config demonstrates SetStatusBarDisable, SetBackKeyDisable, SetVolumeButonDisable and SetRecentAppDisable.
 - Second config demonstrates SetHomeKeyDisable
 - The related ExitPasswordEnabled / ExitPasswordValue settings are also demonstrated.



Summary of options

Option	Description
	Managed Android devices. The direction of the industry and most compliant with the Android ecosystem. Continue to monitor how it is realized on Zebra's devices for your enterprise deployment.
	Mobility eXtensions (MX) Recommended way to secure and lock down your device. Can be used independently or in conjunction to supplement other lock down tools such as EHS, EB or another third party tool.
	Enterprise Home Screen (EHS) Great way to have a customized launcher in your Enterprise deployment and offers some easy to use lock down features. Should be used in conjunction with MX for a complete solution.
	Enterprise Browser (EB) If you are planning on deploying an EB application then the new CustomKioskMode will be useful and easy to switch on. Should be used in conjunction with MX for a complete solution.

Resources

- Related Blogs:
 - <https://developer.zebra.com/community/android/android-forums/android-blogs/blog/2017/04/10/locking-down-your-device>
- Sample Apps shown in this presentation:
 - <https://github.com/darryncampbell/MxLockDown>
 - <https://github.com/darryncampbell/LockTaskMode-Exerciser>
- YouTube videos referenced in this presentation:
 - <https://www.youtube.com/watch?v=Za0OQo8DRM4>
- Related talks
 - <https://developer.zebra.com/community/developer-events/blog/2017/01/27/dev-talk-feb-22-best-enterprise-features-in-android-m-n>

Questions?