Zebra Developers

Build Your Edge

Writing Apps for the Work Profile

Darryn Campbell
SW Architect, Zebra Technologies
@darryncampbell

October 20th, 2021

Writing Apps for the Work Profile

Who am I?

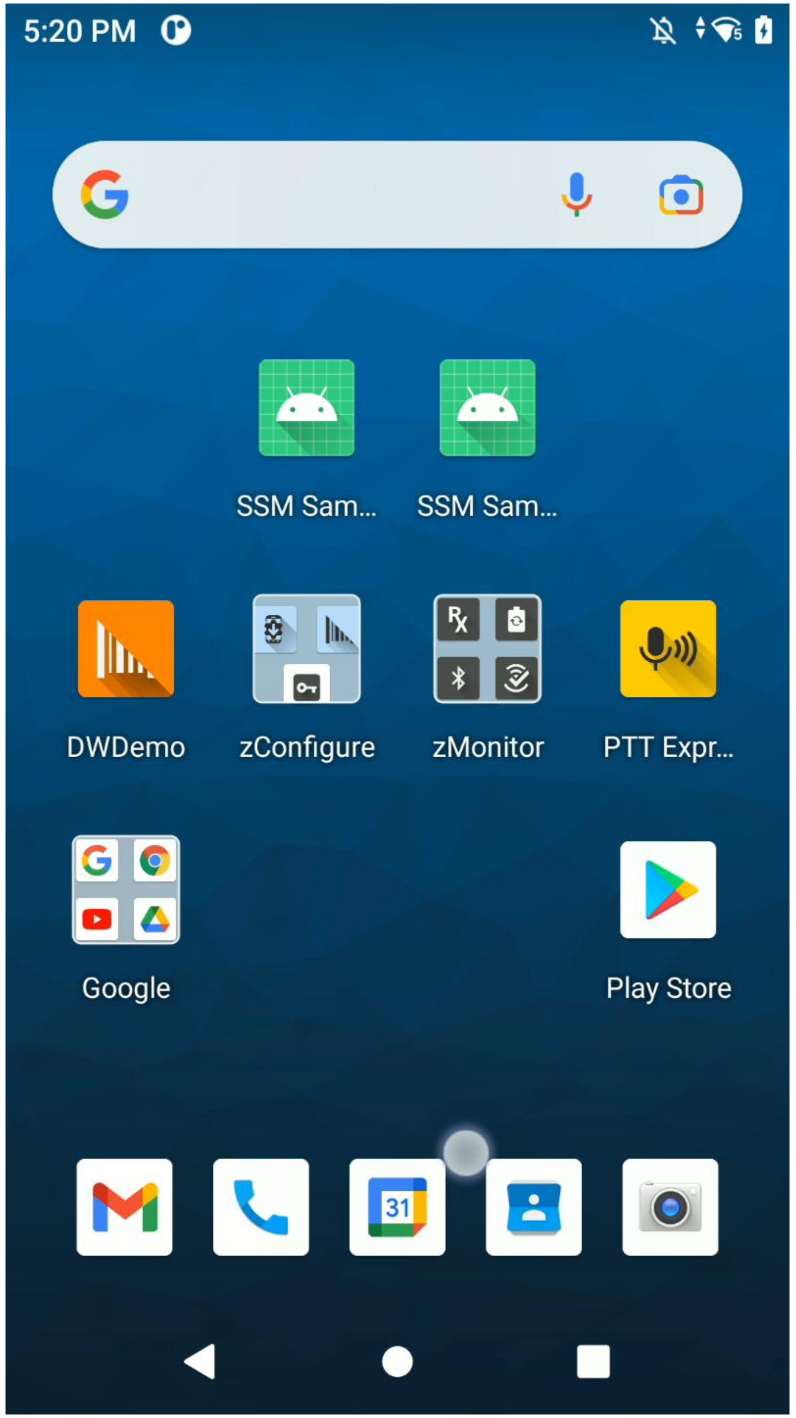
- Developer advocate / Software architect for Zebra Technologies
  - Android OEM developing task specific devices
- Responsible for our Android developer kits & APIs

@darryncampbell | darryncampbell.co.uk

Writing Apps for the Work Profile

What is the Android work profile?

- What is the Android Work Profile?
- First introduced in Android Lollipop
- Separate, sandboxed profile (user) on Android devices
- Work apps and associated data
- “Owned” by an EMM. Remotely managed and provisioned
- Separate Play Store



Writing Apps for the Work Profile

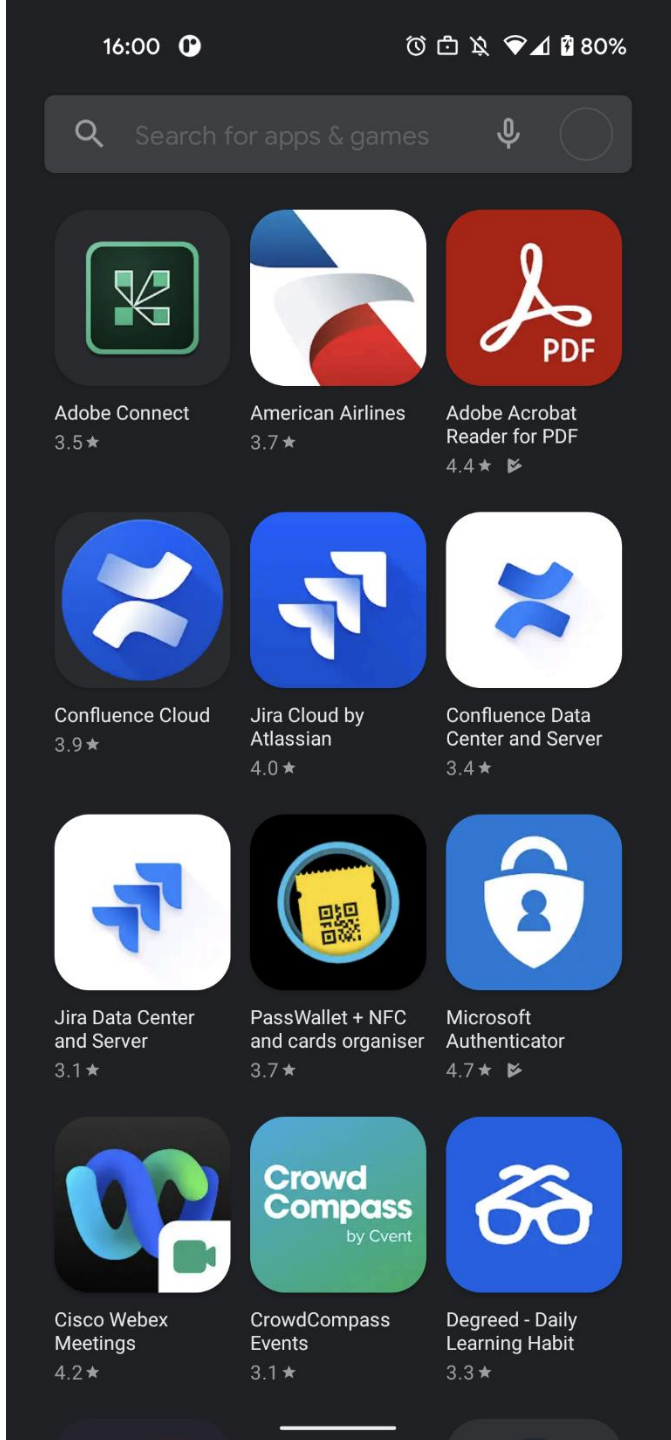
Should you support the Android Work Profile?

- Android for everyone: Casting the widest net
- Who are your users?
- Could your application be used in the workplace?
- Is an organization likely to “approve” your app for their curated Play Store?

Writing Apps for the Work Profile

Remotely Provisioning apps into the Work Profile

1. Organization select an EMM provider:
   - Android Enterprise Essentials
   - Google Advanced Endpoint Management
   - AER EMM Provider (currently 139 of these)
2. End users download the EMM application from the Play Store
   - Many AER EMMs will also support iOS
3. End users sign into EMM, e.g. SSO or Federated ID
4. "Required" Applications are provisioned to the user
5. User may select additional applications from a curated list via the Managed Play Store.



Writing Apps for the Work Profile

Writing apps for the Work Profile: Configuration

- Consider **how** you configure your app today:
  - How do you expose application configuration?
  - Would you still want the app to be configured in the work profile?
- Consider **what** you configure in your app today:
  - Would the configuration be different in the work profile?
  - Are there any features of your app you might want to disable when running in the work profile?
  - Does my app collect data that an Enterprise device administrator might consider proprietary?

Writing Apps for the Work Profile

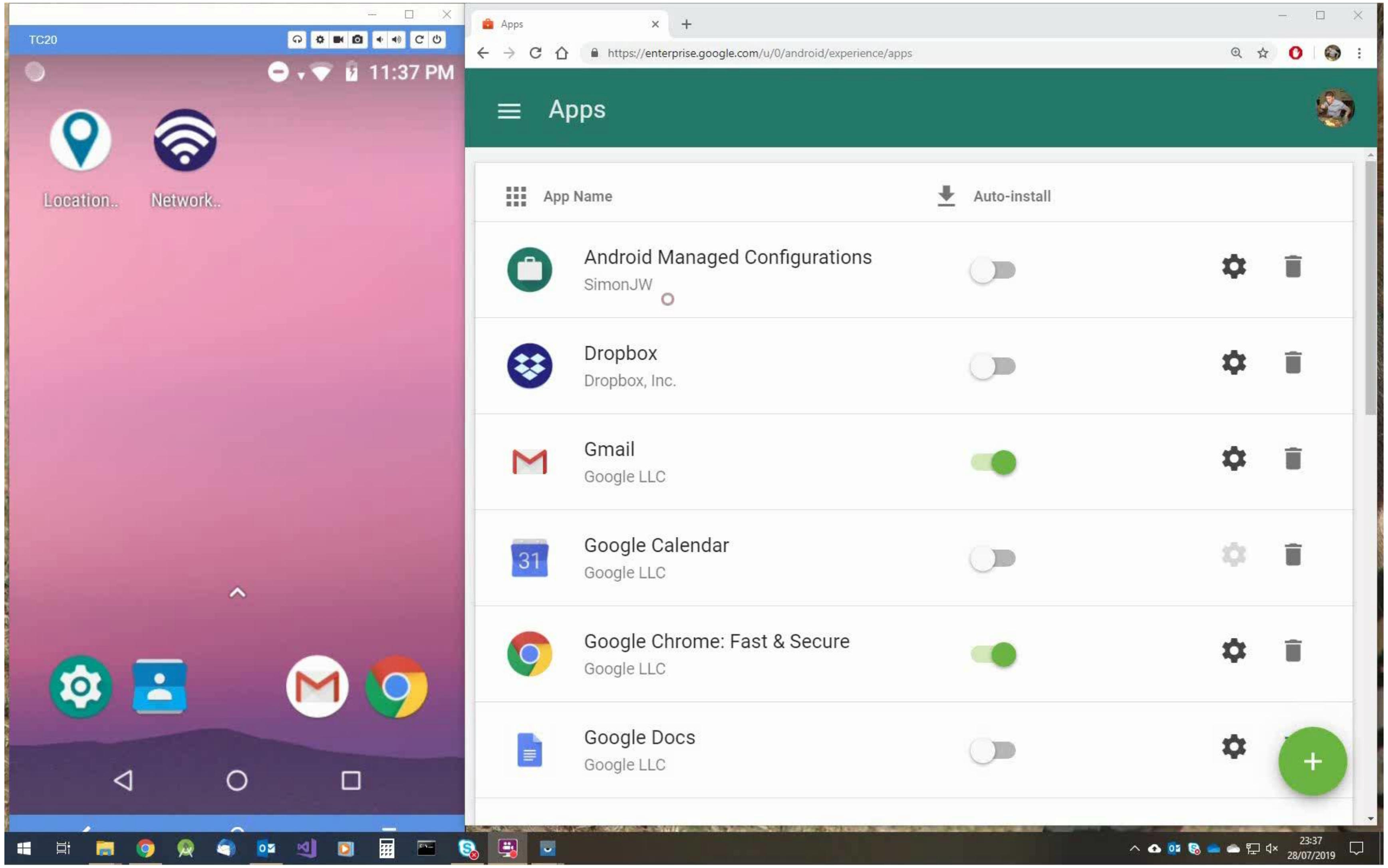
Writing apps for the Work Profile: Managed Configurations

1. Define your application configurations (application restrictions) in XML
2. Supported types: Bool, String, Integer, Choice, Bundle
3. Check state of managed configs:
   - OnResume()
   - Listen for broadcast
4. Configure the application as appropriate

```
<resources>
   <!-- Bool restriction -->
   <string name="title_can_say_hello">Can say hello</string>
   <string name="desc_can_say_hello">Whether app can say Hello</string>
   <bool name="default_can_say_hello">false</bool>
</resources>
```

Writing Apps for the Work Profile

Writing apps for the Work Profile: Managed Configurations



https://enterprise.google.com/u/0/android/experience/apps

Apps

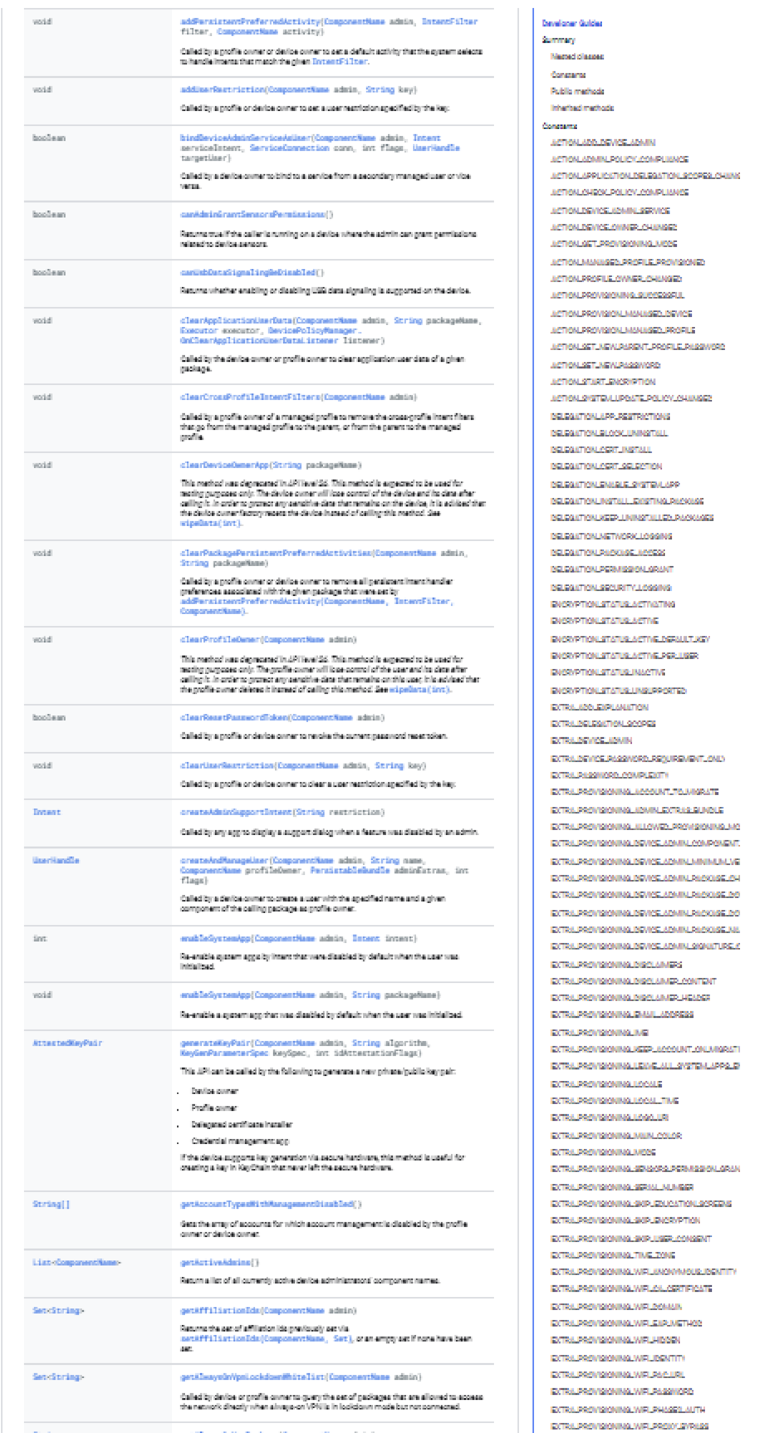
App Name	Auto-install
Android Managed Configurations SimonJW	Off
Dropbox Dropbox, Inc.	Off
Gmail Google LLC	On
Google Calendar Google LLC	Off
Google Chrome: Fast & Secure Google LLC	On
Google Docs Google LLC	Off

@darryncampbell | darryncampbell.co.uk

Writing Apps for the Work Profile

Other development considerations

- The user may frequently disable the work profile, stopping your app
- Will your application be present in both work and non-work profile?
- The behaviour of the device may be locked down in unexpected ways
  - See Device Policy Manager for a full capabilities exposed to EMMs →
  - Examples:
    - Disable account sign-in
    - Disable camera
    - Disable backup
    - Disable location
    - Share contacts & phone numbers between profiles
    - Share calendars between profiles
    - & many more



void	addPersistentPreferredActivity(ComponentName admin, IntentFilter filter, ComponentName activity) Called by a profile owner or device owner to set a default activity that the system selects to handle intents that match the given IntentFilter.
void	addUserRestriction(ComponentName admin, String key) Called by a profile or device owner to set a user restriction specified by the key.
boolean	bindDeviceAdminServiceAsUser(ComponentName admin, Intent serviceIntent, ServiceConnection conn, int flags, UserHandle targetUser) Called by a device owner to bind to a service from a secondary managed user or vice versa.
boolean	canAdminGrantSensorsPermissions() Returns true if the caller is running on a device where the admin can grant permissions related to device sensors.
boolean	canUsbDataSignalingBeDisabled() Returns whether enabling or disabling USB data signaling is supported on the device.
void	clearApplicationUserData(ComponentName admin, String packageName, Executor executor, DevicePolicyManager.OnClearApplicationUserDataListener listener) Called by the device owner or profile owner to clear application user data of a given package.
void	clearCrossProfileIntentFilters(ComponentName admin) Called by a profile owner of a managed profile to remove the cross-profile intent filters that go from the managed profile to the parent, or from the parent to the managed profile.
void	clearDeviceOwnerApp(String packageName) This method was deprecated in API level 26. This method is expected to be used for testing purposes only. The device owner will lose control of the device and its data after calling it. In order to protect any sensitive data that remains on the device, it is advised that the device owner factory resets the device instead of calling this method. See wipeData(int).
void	clearPackagePersistentPreferredActivities(ComponentName admin, String packageName) Called by a profile owner or device owner to remove all persistent intent handler preferences associated with the given package that were set by addPersistentPreferredActivity(ComponentName, IntentFilter, ComponentName).
void	clearProfileOwner(ComponentName admin) This method was deprecated in API level 26. This method is expected to be used for testing purposes only. The profile owner will lose control of the user and its data after calling it. In order to protect any sensitive data that remains on this user, it is advised that the profile owner deletes it instead of calling this method. See wipeData(int).
boolean	clearResetPasswordToken(ComponentName admin) Called by a profile or device owner to revoke the current password reset token.
void	clearUserRestriction(ComponentName admin, String key) Called by a profile or device owner to clear a user restriction specified by the key.
Intent	createAdminSupportIntent(String restriction) Called by any app to display a support dialog when a feature was disabled by an admin.
UserHandle	createAndManageUser(ComponentName admin, String name, ComponentName profileOwner, PersistableBundle adminExtras, int flags) Called by a device owner to create a user with the specified name and a given component of the calling package as profile owner.
int	enableSystemApp(ComponentName admin, Intent intent) Re-enable system apps by intent that were disabled by default when the user was initialized.
void	enableSystemApp(ComponentName admin, String packageName) Re-enable a system app that was disabled by default when the user was initialized.
AttestedKeyPair	generateKeyPair(ComponentName admin, String algorithm, KeyGenParameterSpec keySpec, int idAttestationFlags) This API can be called by the following to generate a new private/public key pair: • Device owner • Profile owner • Delegated certificate installer • Credential management app If the device supports key generation via secure hardware, this method is useful for creating a key in KeyChain that never left the secure hardware.
String[]	getAccountTypesWithManagementDisabled() Gets the array of accounts for which account management is disabled by the profile owner or device owner.
List<ComponentName>	getActiveAdmins() Return a list of all currently active device administrators' component names.
Set<String>	getAffiliationIds(ComponentName admin) Returns the set of affiliation ids previously set via setAffiliationIds(ComponentName, Set), or an empty set if none have been set.
Set<String>	getAlwaysOnVpnLockdownWhitelist(ComponentName admin) Called by device or profile owner to query the set of packages that are allowed to access the network directly when always-on VPN is in lockdown mode but not connected.

Developer Guides

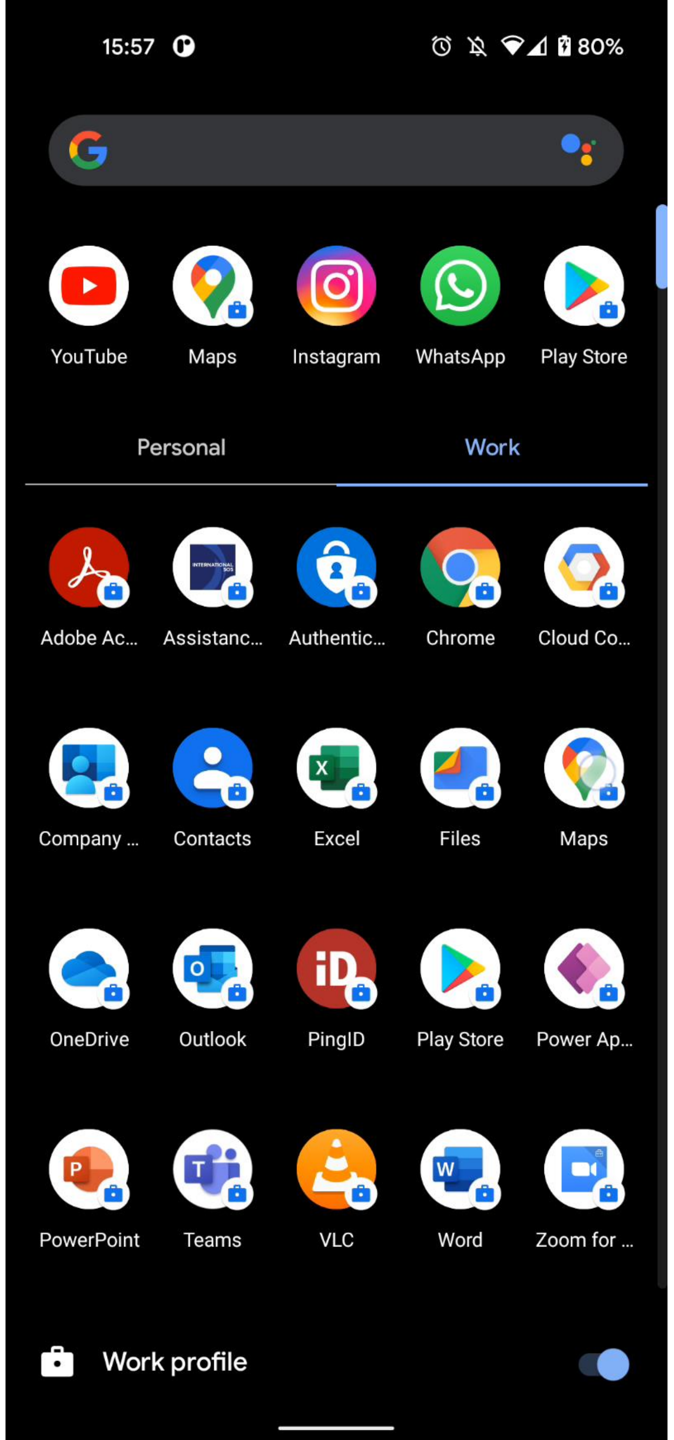
Summary
- Nested classes
- Constants
- Public methods
- Inherited methods

Constants
- ACTION_ADD_DEVICE_ADMIN
- ACTION_ADMIN_POLICY_COMPLIANCE
- ACTION_APPLICATION_DELEGATION_SCOPES_CHANG
- ACTION_CHECK_POLICY_COMPLIANCE
- ACTION_DEVICE_ADMIN_SERVICE
- ACTION_DEVICE_OWNER_CHANGED
- ACTION_GET_PROVISIONING_MODE
- ACTION_MANAGED_PROFILE_PROVISIONED
- ACTION_PROFILE_OWNER_CHANGED
- ACTION_PROVISIONING_SUCCESSFUL
- ACTION_PROVISION_MANAGED_DEVICE
- ACTION_PROVISION_MANAGED_PROFILE
- ACTION_SET_NEW_PARENT_PROFILE_PASSWORD
- ACTION_SET_NEW_PASSWORD
- ACTION_START_ENCRYPTION
- ACTION_SYSTEM_UPDATE_POLICY_CHANGED
- DELEGATION_APP_RESTRICTIONS
- DELEGATION_BLOCK_UNINSTALL
- DELEGATION_CERT_INSTALL
- DELEGATION_CERT_SELECTION
- DELEGATION_ENABLE_SYSTEM_APP
- DELEGATION_INSTALL_EXISTING_PACKAGE
- DELEGATION_KEEP_UNINSTALLED_PACKAGES
- DELEGATION_NETWORK_LOGGING
- DELEGATION_PACKAGE_ACCESS
- DELEGATION_PERMISSION_GRANT
- DELEGATION_SECURITY_LOGGING
- ENCRYPTION_STATUS_ACTIVATING
- ENCRYPTION_STATUS_ACTIVE
- ENCRYPTION_STATUS_ACTIVE_DEFAULT_KEY
- ENCRYPTION_STATUS_ACTIVE_PER_USER
- ENCRYPTION_STATUS_INACTIVE
- ENCRYPTION_STATUS_UNSUPPORTED
- EXTRA_ADD_EXPLANATION
- EXTRA_DELEGATION_SCOPES
- EXTRA_DEVICE_ADMIN
- EXTRA_DEVICE_PASSWORD_REQUIREMENT_ONLY
- EXTRA_PASSWORD_COMPLEXITY
- EXTRA_PROVISIONING_ACCOUNT_TO_MIGRATE
- EXTRA_PROVISIONING_ADMIN_EXTRAS_BUNDLE
- EXTRA_PROVISIONING_ALLOWED_PROVISIONING_MO
- EXTRA_PROVISIONING_DEVICE_ADMIN_COMPONENT_
- EXTRA_PROVISIONING_DEVICE_ADMIN_MINIMUM_VE
- EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_CH
- EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DO
- EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DO
- EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_NA
- EXTRA_PROVISIONING_DEVICE_ADMIN_SIGNATURE_C
- EXTRA_PROVISIONING_DISCLAIMERS
- EXTRA_PROVISIONING_DISCLAIMER_CONTENT
- EXTRA_PROVISIONING_DISCLAIMER_HEADER
- EXTRA_PROVISIONING_EMAIL_ADDRESS
- EXTRA_PROVISIONING_IMEI
- EXTRA_PROVISIONING_KEEP_ACCOUNT_ON_MIGRATI
- EXTRA_PROVISIONING_LEAVE_ALL_SYSTEM_APPS_E
- EXTRA_PROVISIONING_LOCALE
- EXTRA_PROVISIONING_LOCAL_TIME
- EXTRA_PROVISIONING_LOGO_URI
- EXTRA_PROVISIONING_MAIN_COLOR
- EXTRA_PROVISIONING_MODE
- EXTRA_PROVISIONING_SENSORS_PERMISSION_GRAN
- EXTRA_PROVISIONING_SERIAL_NUMBER
- EXTRA_PROVISIONING_SKIP_EDUCATION_SCREENS
- EXTRA_PROVISIONING_SKIP_ENCRYPTION
- EXTRA_PROVISIONING_SKIP_USER_CONSENT
- EXTRA_PROVISIONING_TIME_ZONE
- EXTRA_PROVISIONING_WIFI_ANONYMOUS_IDENTITY
- EXTRA_PROVISIONING_WIFI_CA_CERTIFICATE
- EXTRA_PROVISIONING_WIFI_DOMAIN
- EXTRA_PROVISIONING_WIFI_EAP_METHOD
- EXTRA_PROVISIONING_WIFI_HIDDEN
- EXTRA_PROVISIONING_WIFI_IDENTITY
- EXTRA_PROVISIONING_WIFI_PAC_URL
- EXTRA_PROVISIONING_WIFI_PASSWORD
- EXTRA_PROVISIONING_WIFI_PHASE2_AUTH
- EXTRA_PROVISIONING_WIFI_PROXY_BYPASS

Writing Apps for the Work Profile

Other development considerations

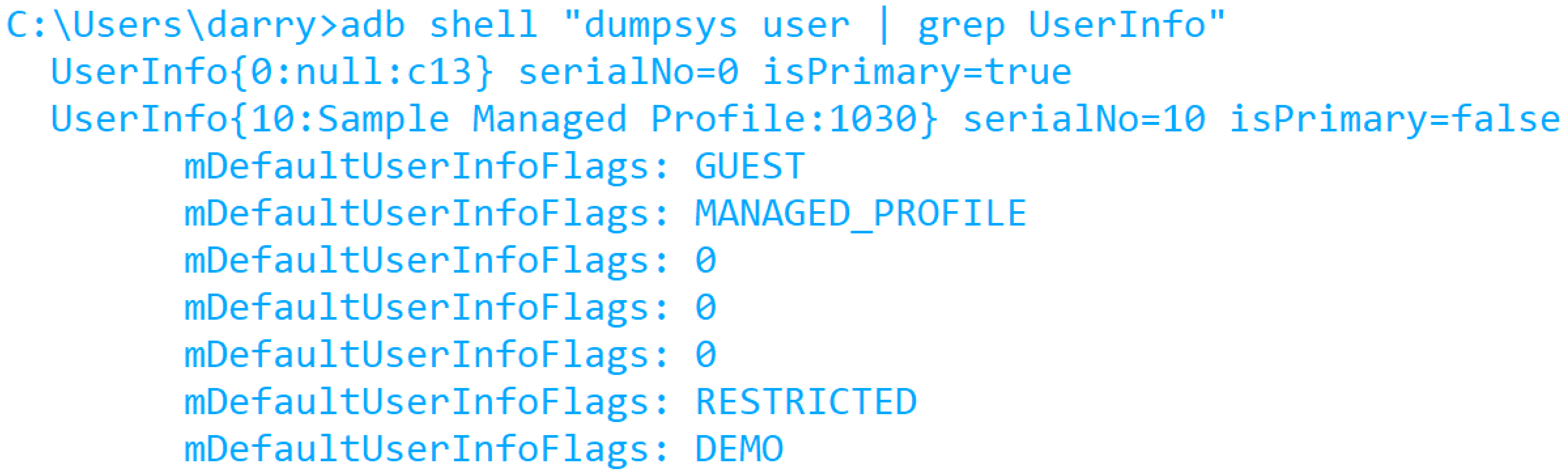
- The user may frequently disable the work profile, stopping your app
- Will your application be present in both work and non-work profile?
- The behaviour of the device may be locked down in unexpected ways
  - See Device Policy Manager for a full capabilities exposed to EMMs →
  - Examples:
    - Disable account sign-in
    - Disable camera
    - Disable backup
    - Disable location
    - Share contacts & phone numbers between profiles
    - Share calendars between profiles
    - & many more



Writing Apps for the Work Profile

Testing your app

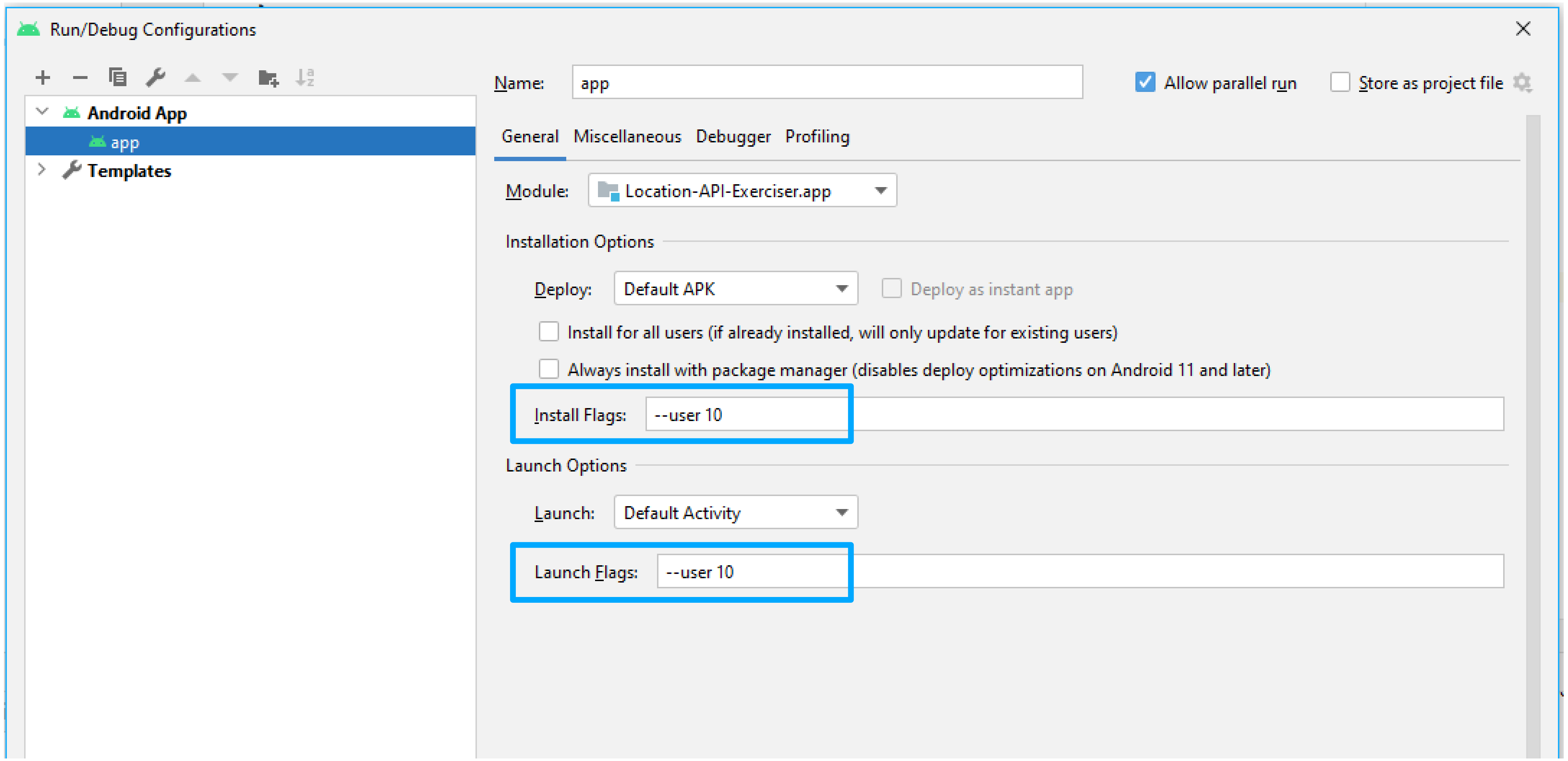
```
adb shell “dumpsys user | grep UserInfo”
```



```
C:\Users\darry>adb shell "dumpsys user | grep UserInfo"
  UserInfo{0:null:c13} serialNo=0 isPrimary=true
  UserInfo{10:Sample Managed Profile:1030} serialNo=10 isPrimary=false
        mDefaultUserInfoFlags: GUEST
        mDefaultUserInfoFlags: MANAGED_PROFILE
        mDefaultUserInfoFlags: 0
        mDefaultUserInfoFlags: 0
        mDefaultUserInfoFlags: 0
        mDefaultUserInfoFlags: RESTRICTED
        mDefaultUserInfoFlags: DEMO
```

Writing Apps for the Work Profile

Testing your app



Run/Debug Configurations

- Android App
  - app
- Templates

Name: app

[✓] Allow parallel run
[ ] Store as project file

General | Miscellaneous | Debugger | Profiling

Module: Location-API-Exerciser.app

Installation Options

Deploy: Default APK
[ ] Deploy as instant app
[ ] Install for all users (if already installed, will only update for existing users)
[ ] Always install with package manager (disables deploy optimizations on Android 11 and later)

Install Flags: --user 10

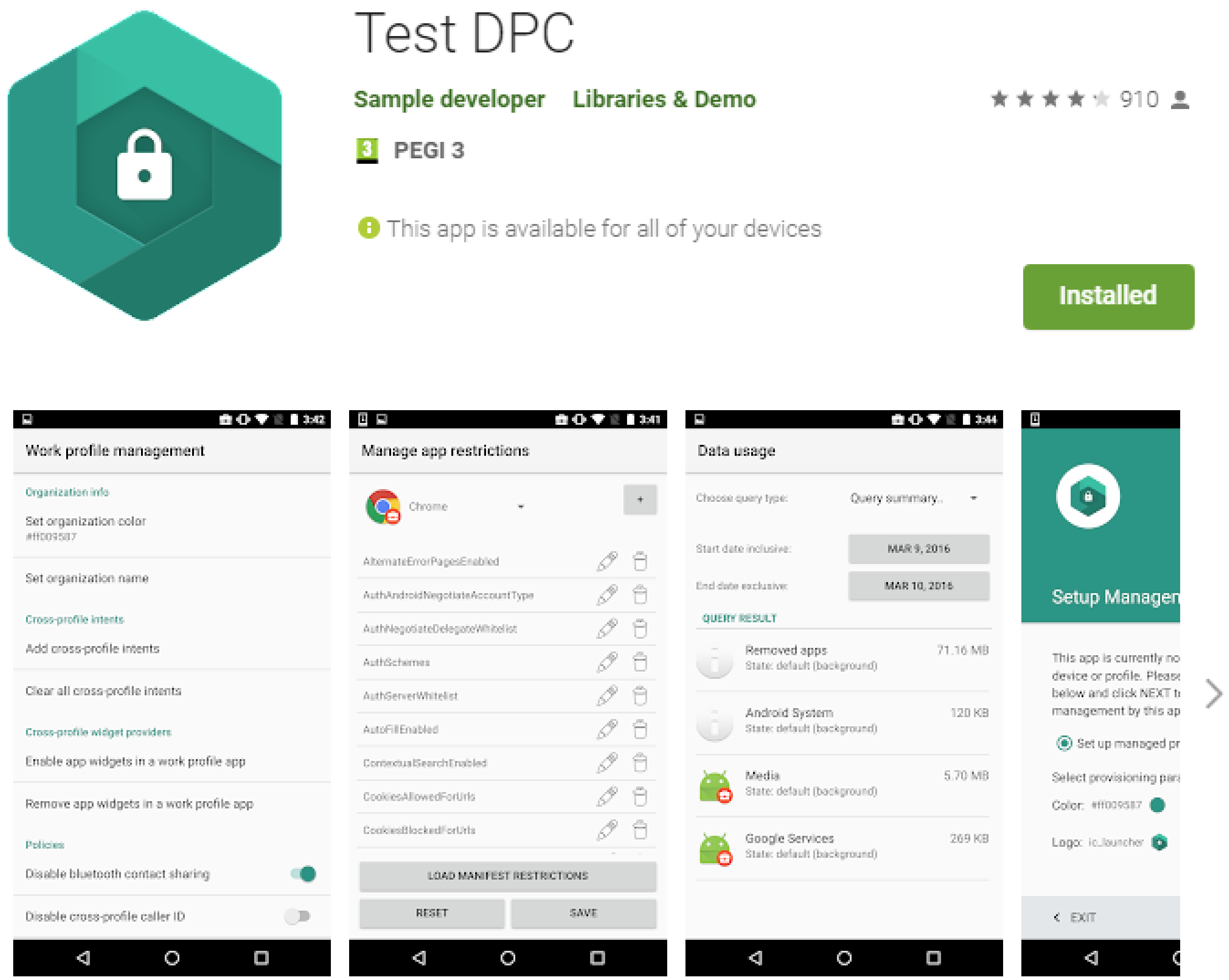
Launch Options

Launch: Default Activity

Launch Flags: --user 10

Writing Apps for the Work Profile

Testing your app: Test DPC



Test DPC

Sample developer Libraries & Demo

★★★★☆ 910

PEGI 3

This app is available for all of your devices

Installed

Work profile management

Organization info

Set organization color
#ff009587

Set organization name

Cross-profile intents

Add cross-profile intents

Clear all cross-profile intents

Cross-profile widget providers

Enable app widgets in a work profile app

Remove app widgets in a work profile app

Policies

Disable bluetooth contact sharing

Disable cross-profile caller ID

Manage app restrictions

Chrome

AlternateErrorPagesEnabled
AuthAndroidNegotiateAccountType
AuthNegotiateDelegateWhitelist
AuthSchemes
AuthServerWhitelist
AutoFillEnabled
ContextualSearchEnabled
CookiesAllowedForUrls
CookiesBlockedForUrls

LOAD MANIFEST RESTRICTIONS

RESET SAVE

Data usage

Choose query type: Query summary..

Start date inclusive: MAR 9, 2016

End date exclusive: MAR 10, 2016

QUERY RESULT

Removed apps State: default (background)	71.16 MB
Android System State: default (background)	120 KB
Media State: default (background)	5.70 MB
Google Services State: default (background)	269 KB

Setup Managen

This app is currently no
device or profile. Please
below and click NEXT t
management by this ap

Set up managed pr

Select provisioning para

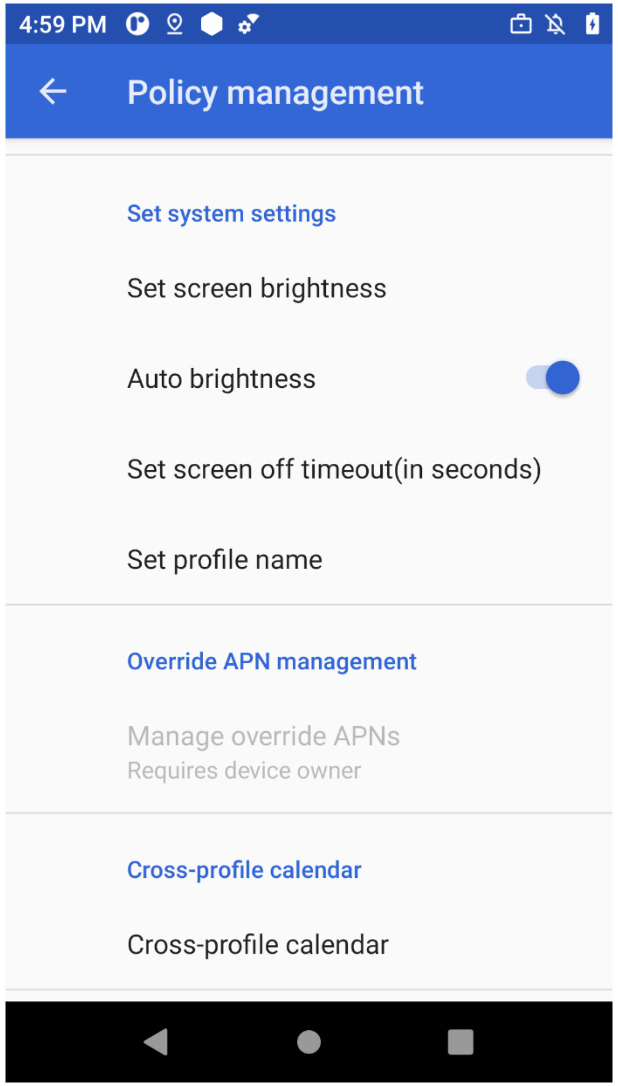
Color: #ff009587

Logo: ic_launcher

EXIT

Writing Apps for the Work Profile

Testing your app: Test DPC


4:59 PM

Policy management

Set system settings

Set screen brightness

Auto brightness

Set screen off timeout(in seconds)

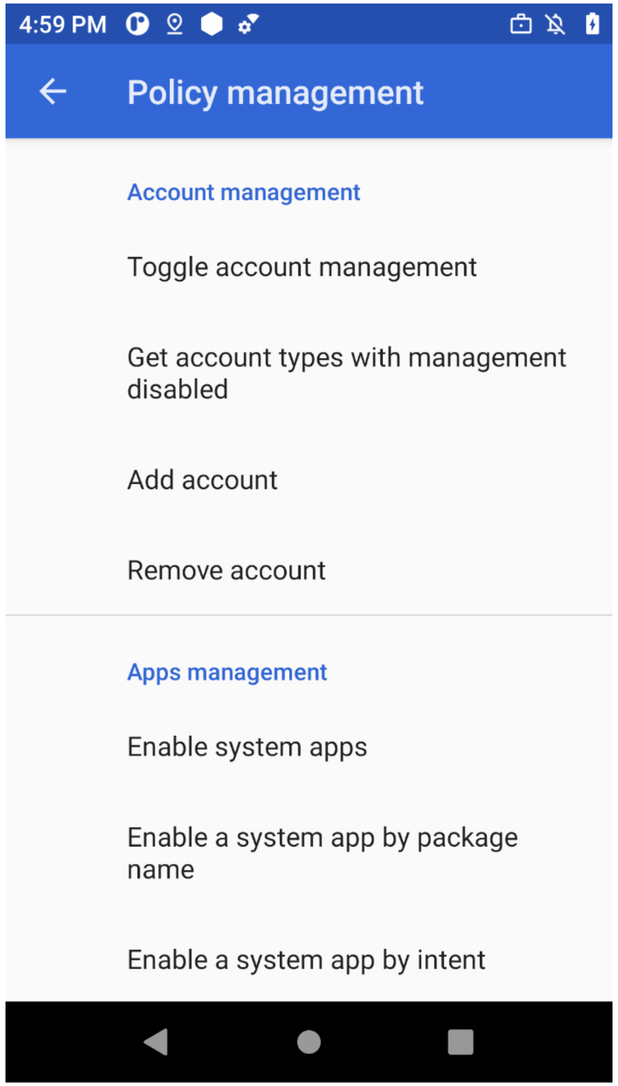
Set profile name

Override APN management

Manage override APNs
Requires device owner

Cross-profile calendar

Cross-profile calendar


4:59 PM

Policy management

Account management

Toggle account management

Get account types with management disabled

Add account

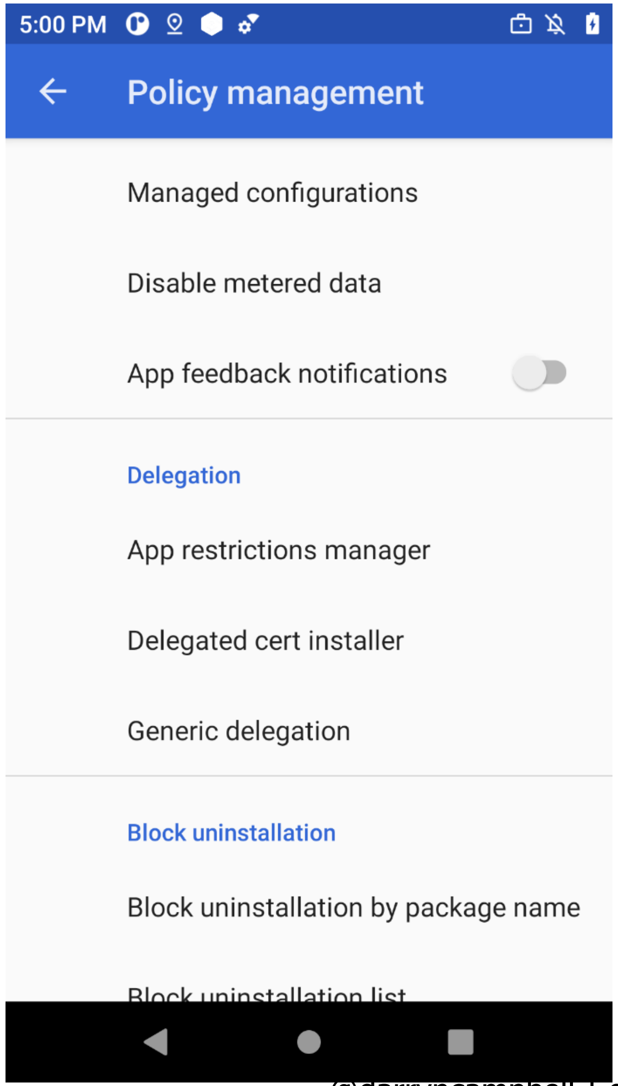
Remove account

Apps management

Enable system apps

Enable a system app by package name

Enable a system app by intent


5:00 PM

Policy management

Managed configurations

Disable metered data

App feedback notifications

Delegation

App restrictions manager

Delegated cert installer

Generic delegation

Block uninstallation

Block uninstallation by package name

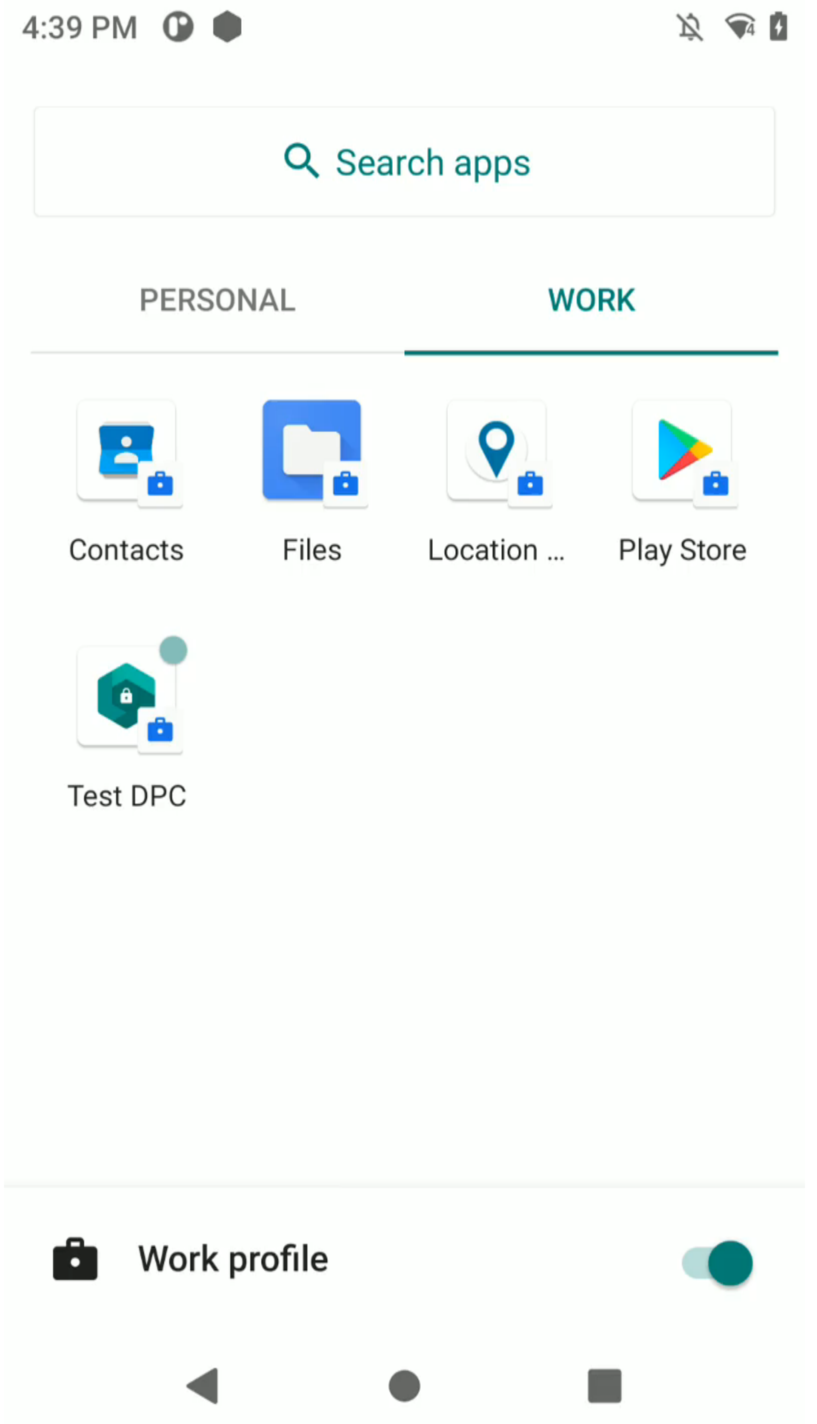
Block uninstallation list

Writing Apps for the Work Profile

Testing your app: Test DPC

Example:

- Application installed into Work Profile which accesses location
- Will request location permission when launched
- In work profile, expect location permission to be pre-granted
- Use Test DPC to specify the pre-approved runtime permissions



Writing Apps for the Work Profile

Resources

- Test DPC: Play Store | Source code
- DevicePolicyManager API: https://developer.android.com/reference/android/app/admin/DevicePolicyManager
- YouTube Play list of videos shown in this presentation: https://youtube.com/playlist?list=PLj8D9Diz5FBrAvULNDvMBa7aZ1FJnvM0_
  - Enabling the work profile: https://youtu.be/Dx3sT_KH1S8
  - Managed configuration example: https://youtu.be/XU9k1rsWRMQ
  - Maps sign-in forbidden: https://youtu.be/sgc_Vm3-lr8
  - Test DPC to allow Runtime permissions: https://youtu.be/if3KxhWR4NE

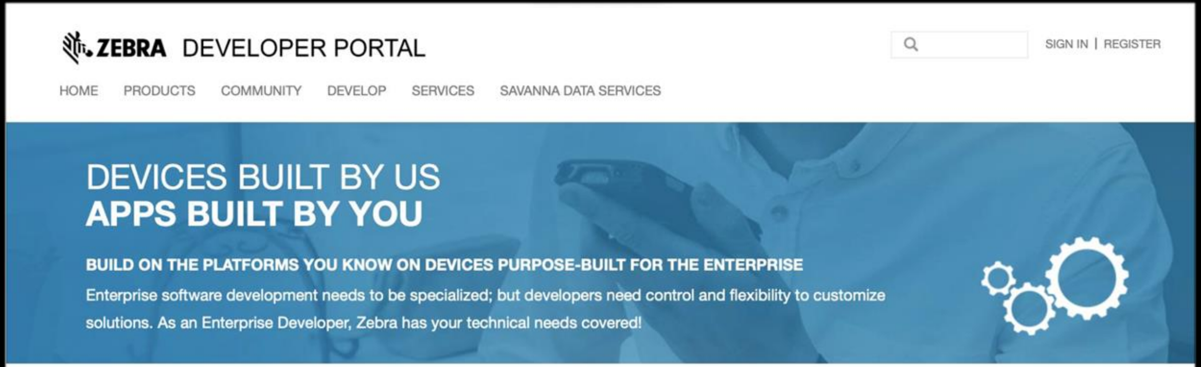
Writing Apps for the Work Profile

Questions?

http://developer.zebra.com

Zebra Developers
Build Your Edge

Be a part of our community


ZEBRA DEVELOPER PORTAL

SIGN IN | REGISTER

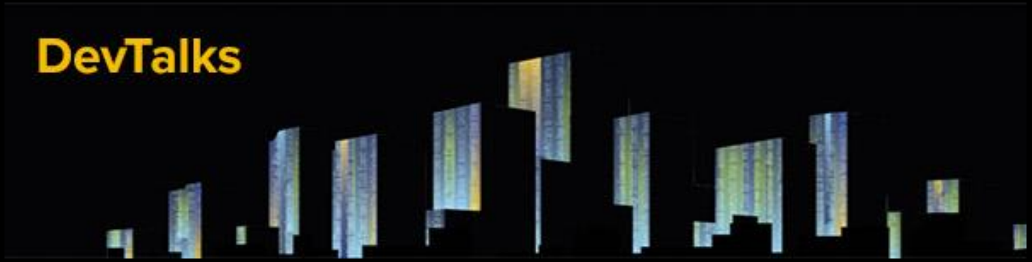
HOME PRODUCTS COMMUNITY DEVELOP SERVICES SAVANNA DATA SERVICES

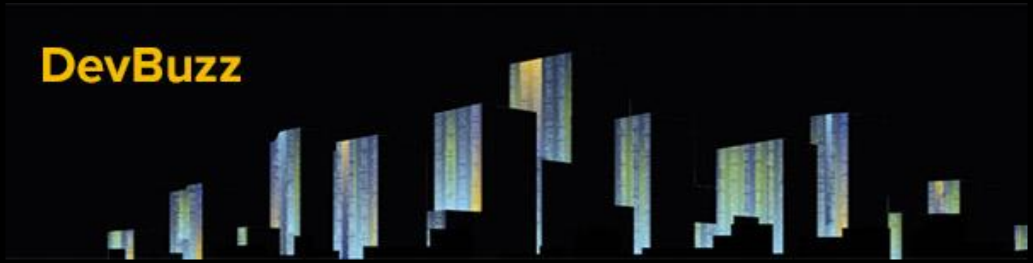
DEVICES BUILT BY US
APPS BUILT BY YOU

BUILD ON THE PLATFORMS YOU KNOW ON DEVICES PURPOSE-BUILT FOR THE ENTERPRISE
Enterprise software development needs to be specialized; but developers need control and flexibility to customize solutions. As an Enterprise Developer, Zebra has your technical needs covered!

Join our Developer Community

ZEBRA DEVELOPER PORTAL
http://developer.zebra.com





Follow Zebra!

Thank You