



# Integrating a Mobile Device Management Solution in Android

Laura Kanerva

BACHELOR'S THESIS  
November 2021

Degree Programme in Business Information Systems  
Option of Software Development

## ABSTRACT

Tampereen ammattikorkeakoulu  
Tampere University of Applied Sciences  
Degree Programme in Business Information Systems  
Option of Software Development

Laura Kanerva:  
Integrating a Mobile Device Management Solution in Android

Bachelor's thesis 43 pages  
November 2021

---

Employers are embracing the *bring your own device* (BYOD) phenomenon and its benefits. Giving the workforce access to enterprise data anytime and anywhere with their own devices gives them freedom and boosts productivity and profitability. However, it also creates more exposure to security threats. If the process for BYOD is not managed properly, sensitive data could be exposed outside of the trusted network. A critical aspect of protecting data on mobile devices as part of a BYOD program is mobile device management (MDM). It provides organizations with end-to-end security and the ability to remotely monitor, control, and manage enrolled mobile devices.

The objective of this thesis was to investigate the features of an MDM solution and to find out how much time and resources it requires to integrate such a solution into an Android application. The purpose was to create a prototype to test the integration process and to find potential problem areas. This was done by integrating VMware Workspace ONE MDM into the M-Files Android mobile application, using the Workspace ONE software development kit (SDK). A literature review was conducted to gain a better understanding of mobile device management, bring your own device and mobile security. The VMware Workspace ONE product documentation was also explored and utilised in the integration process.

Doing research, creating a prototype, and testing it before committing to an MDM solution is recommended. Considerable amount of work will need to be done to bring the integration to a level where it is ready for release. The library dependencies and the Workspace ONE SDK update frequently, which easily leads to compatibility issues. Further research and testing are required to achieve the best possible outcome of the Workspace ONE integration.

---

Key words: mobile device management, bring your own device, mobile security, android, software development kit

## CONTENTS

1	INTRODUCTION .....	5
2	ANDROID OPERATING SYSTEM.....	7
3	MOBILE DEVICE SECURITY .....	8
3.1	Bring Your Own Device .....	8
3.2	Security threats .....	9
3.2.1	External attackers.....	10
3.2.2	Internal attackers .....	11
3.2.3	Rooting .....	12
3.2.4	Loss and theft.....	12
3.2.5	Malware.....	13
3.2.6	Lack of good authentication.....	14
3.2.7	Lack of security awareness .....	15
4	MOBILE DEVICE MANAGEMENT .....	16
4.1	Mobile device management policies .....	16
4.2	Features .....	18
4.2.1	Controlling applications .....	18
4.2.2	Protecting the device from loss and theft.....	19
5	INTEGRATION PROCESS .....	21
5.1	Workspace ONE .....	21
5.1.1	Device deployment.....	21
5.1.2	Configurations .....	26
5.1.3	Distributing applications.....	28
5.2	Integrating Workspace ONE into Android application .....	30
5.2.1	Integration preparation .....	30
5.2.2	Framework level integration .....	31
5.2.3	Testing and troubleshooting .....	36
6	DISCUSSION .....	40
	REFERENCES .....	42

**ABBREVIATIONS AND TERMS**

adb	Android Debug Bridge
APK	Android Application Package
BYOD	Bring Your Own Device
EMM	Enterprise Mobility Management
GPS	Global Positioning System
IT	Information Technology
MDM	Mobile Device Management
OS	Operating System
OTA	Over-The-Air
PIN	Personal Identification Number
SDK	Software Development Kit
SSO	Single Sign-On
UEM	Unified Endpoint Manager
USB	Universal Serial Bus

## 1 INTRODUCTION

Mobile phones were once only used for making phone calls and sending text messages. Modern smartphones, however, are closer to handheld computers that enable people to do anything from shopping online and making video calls to banking and creating documents. Just like computers, mobile devices also use operating systems (OS) that manage their memory and resources, bring new advanced functions - and can be exploited. (Mobile operating systems 2021)

In many ways, smartphones are more vulnerable than laptops and desktop computers. Many computer users know that their computer can be attacked and get infected, and maintain a decent level of security on their device. Unfortunately, the same does not go for mobile phones, that are all too often left without any protection. Mobile devices and software are frequently sold and distributed with little to no documentation, as they are meant to be easily operated, and users are expected to do their own research. This leads to users having major gaps in their security knowledge. (Dunham 2009; Thompson, McGill & Wang 2017)

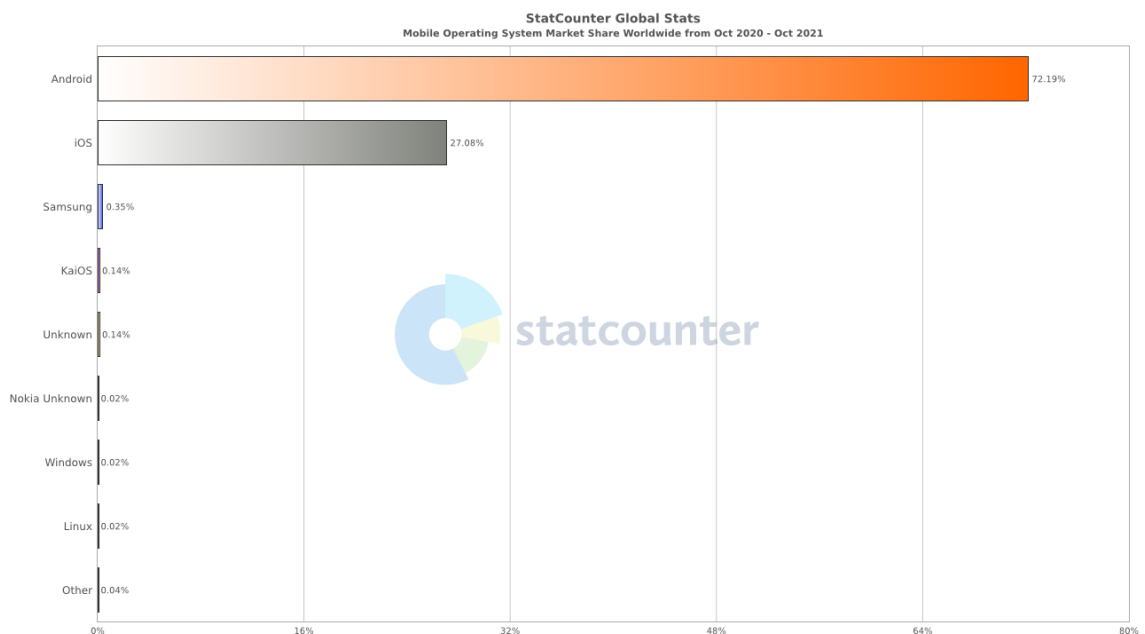
The number of mobile devices operating worldwide keeps increasing every year. While these devices have not completely replaced computers, they have grown in such numbers that organizations cannot refuse the requests for adoption of bring your own device (BYOD) or stop unauthorized devices from connecting to the corporate network. Employees expect to be able to work anytime, anywhere, and on any device. Suddenly, business information assets are being stored together with personal data on these devices, and companies have less control over them than before. The new era of mobility requires that the devices are being secured, thus enabling the workforce to be more successful, instead of trying to stop the use of them. (Lim, Coolidge & Hourani 2013)

With the increasing popularity of BYOD comes the excruciating issue of enforcing corporate compliance on mobile devices. Employers can install monitoring, troubleshooting, and security applications on company-owned devices, but uncontrolled and unmonitored employee-owned devices are left without proper supervision. Administrators cannot control system upgrades or installed applications

on mobile devices as easily as they have done in the desktop environment. Employing a mobile device management (MDM) solution effectively reduces the risks associated with this space. MDM helps to enforce administrative policies and provides the tools for ensuring that all devices connected to the company network are compliant and safe. (Campagna, Iyer & Krishnan 2011; Bergman, Stanfield, Rouse & Scambray 2013)

## 2 ANDROID OPERATING SYSTEM

A mobile operating system (OS) is the software that allows mobile devices to run applications and other programs. Operating system manages the memory and resources of the mobile device and works as a platform for developers to create mobile applications. Two of the biggest mobile operating systems are Android and iOS. In 2021, Android has a bigger market share due to its availability on devices from multiple manufacturers, opposed to iOS that only runs on Apple products (Picture 1). (Mobile operating systems 2021; Poetker 2021)



PICTURE 1. Android has the biggest mobile operating system market share worldwide (GlobalStats statcounter 2021).

The Android operating system is open source, which means that anyone can access the code and make alterations to it. Unfortunately, this can make it easier for attackers to find exploits and use them to their benefit. The open-source nature of Android puts Android devices at greater risk and makes them a lot more vulnerable to attacks than iOS devices. A massive amount of malware applications have found their way onto Android systems, with more expected in the future. (Campagna et al. 2011; Speed, Nykamp, Anderson & Nampalli 2013)

### **3 MOBILE DEVICE SECURITY**

One of the greatest concerns for mobile devices is security. All data is inherently at greater risk while on a mobile device, because mobile devices tend to be less protected than personal computers. The threats against mobile devices are rapidly increasing at the same pace with the usage of these devices. Mobile device users are also more vulnerable to security threats, as they are often left making independent decisions about how to protect themselves with little understanding of technology. According to Thompson, McGill and Wang (2017), the majority of people are inclined to rate their computing skills as good or excellent, even if only a minority of them has received any information security training. This self-efficacy, the belief these people have about their own abilities, could easily backfire. (Speed et al. 2013; Thompson et al. 2017)

The leakage of sensitive information is one of the biggest risks on mobile. Mobile devices often store sensitive data, and once thieves get access to that information, they can sell it once or many times over, or they can use it for other malicious purposes. Stolen data cannot be pulled back, which is why keeping both personal and company data safe is so important. However, security problems are inevitable, and preventing security breaches is often complicated. Employees are tempted to do things they shouldn't do against their better judgement, and different kinds of new threats are constantly emerging. The world is becoming progressively mobile, and the best way to live in it securely is to embrace, adapt, protect, and manage the hordes of mobile devices. (Campagna et al. 2011; Speed et al. 2013)

#### **3.1 Bring Your Own Device**

Due to the massive growth of mobile devices like smart phones and tablets, the popularity of the Bring Your Own Device (BYOD) movement has also risen. BYOD is the trend of companies allowing employees to bring their own devices and use them for work. BYOD has become a big part of the work culture, and mobile devices are now accepted as essential tools in businesses. (Lim et al. 2013)



Some of the key reasons for an organization to adopt BYOD are that it reduces expenses and increases productivity. Companies are saving money in hardware, telecommunications, insurance, and training. Employees buy their own devices and do not need to be trained to use them. Employees can also work whenever and wherever, and are increasingly satisfied with their working conditions. (Zahadat, Blessner, Blackburn & Olson 2015)

Mobile devices that are used for working can either be company-owned or employee-owned. Employees using personally-owned devices to connect to the corporate network introduce new security risks to companies. Personal data gets mixed with sensitive business data on these devices, and employers have very little control over this. From a security point of view, this is not an ideal situation. (Speed et al. 2013)

The BYOD process needs to be managed properly to ensure that the sensitive corporate data does not get exposed outside of the trusted network. Many companies will implement policies regarding the corporate use of a mobile device and the responsibilities the employee has over their BYOD device. The use of BYOD may differ from company to company and may also be impacted by various government rules. There is very little law for companies to follow regarding BYOD. Employers will simply do what they can to protect the company, while employees will do their best to protect themselves. (Speed et al. 2013)

### **3.2 Security threats**

As Lim, Coolidge and Hourani have stated, "every threat begins with the attacker." Constantly evolving mobile devices and the ever-growing number of mobile device users attracts hackers. Every new smartphone feature provides hackers a new, different avenue to access the device without permission, intercept communication, spread malware, and steal personal and business information. (Lim et al. 2013; Weichbroth & Łysik 2020)

Different types of attackers crave different things, normally either money, power, or some sensitive information. Sometimes the goal is to obtain information that can be used as a steppingstone to something much more valuable. Perpetrators have a number of approaches to use to get what they want, and their chosen method of operation depends on their skillset and on whether they work from outside or inside of the company. (Chadd 2018)

### **3.2.1 External attackers**

Political hackers, or 'hacktivists', are the kind of hackers who hack out of conviction. They consider themselves as activists and use their skills to expose or attack governments, companies, organizations, and other entities they see as corrupt. Even an individual, who pushes a contradicting agenda, may become their target. Highly skilled hacktivists often operate by launching mass attacks, distributed denial of service (DDoS) attacks, and highly targeted attacks to cause disruption and discomfort to their target. (Lim et al. 2013; Chadd 2018)

Organized crime is also willing to take advantage of the new attack surfaces that mobility has exposed. The goal for cyber criminals is to steal information, data, or ideas. They may use these valuable assets for corporate espionage or simply monetization. Organized crime is active in the mobile device space, for example in the form of specific schemes targeting Android devices. Common examples of these schemes are spyware, ransomware, and adware. Cybercrimes may cause major financial harm, and are known to have been used to fund other forms of serious crime. (Lim et al. 2013; Cyber organized crime activities 2019)

Not all attackers are as proficient, organized, or malicious, though. Script Kiddies are amateur hackers, who usually abuse vulnerabilities that are already known, with scripts that other hackers have written and shared. They may try to hack systems, networks, or websites like other hackers, but with less skills, resources, and persistence. Often their intention is to get attention or play around, just try and see what happens. Script Kiddies are not always seen as big of a threat as other attackers, but according to Verton, the “lack of maturity and organization

may, in fact, make the less capable hackers more dangerous”. (Verton 2001; Sarangam 2021; Steinz & Pulkkinen 2021)

### **3.2.2 Internal attackers**

Internal attackers are an increasingly concerning security threat that needs to be managed tactfully and with the full support of the corporate executives. Working within the targeted organization is the most effective but less talked about way to steal, damage, and destroy sensitive information. Privileged insiders may have access to sensitive information as part of their normal duties, and knowledge of security controls and how to exploit them. (Gonzalez 2015; Zahadat et al. 2015)

Insiders could be part of any one of the other attacker groups (Lim et al. 2013). A hacker could have a job within any field of work, working in any position from low-level employee to an executive. There may also be disgruntled employees, either current or former, who for some reason are not happy and maybe hold a grudge against the company. They could easily use their knowledge and contacts to do some serious harm. Simply anyone can turn hostile and take advantage of an opportunity to gain access to something valuable. (Steinz & Pulkkinen 2021)

Sometimes innocent employees become victims of cyberattacks and social engineering attacks. They may connect an infected device into the company network or put in their login credentials to grant someone access to the systems for the sake of convenience. They could open an attached document in a phishing email or give their password to an attacker who calls them posing as an employee from the corporate help desk. Unwittingly, this unsuspecting employee is helping an attacker, even if they have no malicious motives behind their actions inside the company. (Speed et al. 2013; Gonzalez 2015; Steinz & Pulkkinen 2021)

Internal attackers could also be nonemployees that are granted access to the company’s network for business reasons. Contractors and consultants, for example, have access to valuable information assets of the company. Some companies may also have partners or outsourcers that are trusted with access to the company systems. Insider threats can cause the most significant harm and are

difficult to detect, because all these people have authorized access and more knowledge of the company systems than an external attacker would have. (Lim et al. 2013; Gonzalez 2015)

### **3.2.3 Rooting**

Many manufacturers limit the users' control of the operating system to protect the users and to keep the devices secure. Users cannot generally modify the OS of the phone or add custom changes. The OS restrictions provide a secure user experience and prevent users from accidentally damaging essential software elements. Rooting aims to bypass these limitations and give Android users access to control all functions on their device. Rooting allows users to alter themes and graphics and install custom software on their device. After the rooting process, users can also download applications from any source, and not just from the approved platforms like the Google Play Store. (Speed et al. 2013)

Although rooting is viewed as a way for users to gain more control over their device, it also compromises it. The device becomes more vulnerable to malware and hacking. Rooting makes it easier for a malicious application to gain full access to the device and any data stored on it. As tempting as the promised benefits are, the consequences of rooting may be catastrophic. (Bergman et al. 2013)

### **3.2.4 Loss and theft**

There are many circumstances where mobile devices may change hands. From the security perspective, it does not make much difference whether a mobile device gets lost, stolen, replaced, or destroyed. The primary concern is not the cost of the device, but that whoever is in possession of the device now has full access to it. Mobile devices frequently contain private information and company data that should not be exposed to others. If an attacker gets unrestricted physical access to a device, there is not much to be done to secure the device and the data stored on it. (Campagna et al. 2011)

Mobile devices are designed to be small and portable. As convenient as it is, they can get lost very easily. According to Weichbroth and Łysik (2020), it is twice as likely for a mobile device to get lost than to be taken with malicious intent. However, even lost phones may be picked up by someone who takes advantage of the situation and never returns the device to its rightful owner. (Weichbroth & Łysik 2020)

Mobile devices are very attractive to thieves. They have monetary value, it is easy to take them imperceptibly, and they often hold even more personal information than a computer does. An attacker may target a certain individual or organization and attempt to steal a particular phone. The typical goals of a targeted attack are to collect information that the attacker can use on further attacks, to steal the identity of the victim, or to destroy or sell data gathered. (Dunham 2009)

### **3.2.5 Malware**

One of the most common methods of spreading malware is through mobile applications. There are numerous places from which users can download mobile apps. Google Play is the primary app store for Android devices. Downloading applications from other app providers may be riskier due to the fact that they are not as heavily policed. Many apps can be downloaded free of charge, which encourages people to experiment with new applications continuously. As a result, mobile devices are constantly transforming and being exposed to potential malware. (Campagna et al. 2011)

Malware is malicious software used by attackers for example to collect sensitive information or to take control of a mobile device. This may cause considerable financial harm to the owner of the device, and they may even become the victim of an identity theft. Malware is often hidden in a way that allows the attacker to do their work without the user being aware of it. When user downloads an app, they may be asked to grant it certain permissions related to location, microphone, or camera, for example. Not many people read these permissions closely before allowing them. If a malicious app is given the permission to access the storage of the device, an attacker may collect all the information stored there. User may also

be tricked into trusting a legitimate appearing, malware containing, counterfeit version of a real app. (Speed et al. 2013)

An employee could unknowingly infect a whole corporate network. When an infected device is connected to a computer through a USB port, the malware can replicate itself and quickly spread from one connected computer to a whole protected network. Malware may also spread through text messages, compromised Wi-Fi hotspots, and email attachments. (Speed et al. 2013)

### **3.2.6 Lack of good authentication**

“Controlling access is the basis of all security. The right people should be allowed in, and the wrong people kept out.” (Townsend 2020). Authentication is used to confirm the identity of the person seeking access and making sure they are properly authorized (Townsend 2020). On Android devices, users can authenticate by providing either a PIN, pattern, password, or fingerprint. They also have the option to completely turn off the security on the lock screen of the device.

Considerable number of people have common, easy-to-remember passwords. People also tend to use the same password for as long as possible, and on multiple sites and services. While it can be convenient for the user, it also makes it easy for a hacker to get access to the sensitive information secured by the password. (Speed et al. 2013)

Most users will resist the bare idea of using a password to unlock their smart phone instead of a PIN. This means that the security of the device is essentially reduced to a four-digit guess. PIN code does not even need to be common to be within the attacker’s reach. A shoulder surfing incident will expose the PIN easily and put the device and the identity of its owner at risk. (Lim et al. 2013)

The length of a password is the most critical factor that differentiates strong passwords from the weak. Long, nonsensical and unpredictable passwords are stronger and safer against attacks than short and logically consistent passwords. There are numerous techniques and programs that hackers use to guess users’

login information. These attacks might take some time, but the success rate is still high with simple passwords. (Speed et al. 2013)

### **3.2.7 Lack of security awareness**

Mobile devices are designed to connect wirelessly to available networks, for example via Wi-Fi or Bluetooth, and very few of them are under the company's control. Mobile devices have a variety of interfaces that cater to the demanded always-on connectivity, and every one of these wireless interfaces can be used to attack and compromise these devices. The radio interfaces are widely used, and they enhance the user experience, which is why users often fail to see them as a problem instead of a feature. (Campagna et al. 2011)

Mobile device users often create vulnerabilities due to the lack of security awareness. Most users have little knowledge about mobile security features, and no access to reliable information about the possible consequences of their security choices. Because mobile device users themselves are one of the greatest threats to sensitive data, their proper behavior would greatly minimize the likelihood that a threat event occurs. However, according to Weichbroth and Łysik (2020), "controlling user behavior is considered to be one of the greatest challenges in mobile device security". (Weichbroth et al. 2020; Wu, Moody, Zhang & Lowry 2020)

## **4 MOBILE DEVICE MANAGEMENT**

Mobile Device Management (MDM) plays a critical role in protecting mobile devices as part of a BYOD program. As more personal and corporate data is communicated through and stored on mobile devices, the need to secure corporate data has become more evident. MDM refers to frameworks or solutions that help companies to control, monitor, and manage mobile devices deployed across enterprises. MDM solutions are used to make sure that the employees who use their personal devices to work agree to the company security controls, and that the company data is isolated from the personal data on the device. Many companies use MDM to ensure that only properly configured and secured mobile devices can access their network. (Campagna et al. 2011; Lim et al. 2013; Speed et al. 2013; Zahadat et al. 2015)

MDM solutions are formed from policies and features that mobile device administrators can control and enforce. There are a number of MDM solutions on the market, each having different advantages and disadvantages. MDM products and features change as rapidly as the market environment evolves. The degree of control that an MDM has over the device also depends on the OS and the device model in question. It is important to keep up to date with changes that impact the company environment and choose the solution that best meets the corporate requirements. (Bergman et al. 2013; Lim et al. 2013; Speed et al. 2013)

Given the mobile nature and the pure amount of smartphones and tablets in a company, one of the most important elements of any MDM product is the ability to manage devices over-the-air (OTA). Devices do not need to be physically connected to a machine or network to be managed, and groups of devices can be centrally managed all at once. This saves a great amount of time spent on managing and monitoring these devices. (Campagna et al. 2011)

### **4.1 Mobile device management policies**

MDM policies are significantly important in preventing mobile device security threats and data breaches. It can be difficult to extend the security policies used



in traditional IT implementations to a mobile environment that has very different needs and components. MDM policies help employees better understand mobile security risks and what actions they can take to reduce them. An MDM policy establishes rules for how mobile devices are used and secured within the company, and ensures that the devices still function properly after limiting the use of them. Without consistent and transparent guidelines, the company is left open to various cybersecurity threats. MDM policies should cover both company-owned and employee-owned devices and any personnel who access company data on a mobile device. The main policies include device policies, provisioning policies, monitoring policies, password policies, and application policies. (Campagna et al. 2011; Speed et al. 2013; How to create a mobile device management policy 2021)

Device policies can define both approved devices and unapproved devices. Policy for approved devices applies to all company-owned mobile devices. An unapproved device would be a mobile device that the company neither endorses nor supports. It is possible to impose restrictions on how and when these devices connect to corporate resources, so denying all connectivity to the enterprise network is not necessary. (Campagna et al. 2011)

Provisioning policies define the lifecycle of mobile device management, including setting up configurations, maintaining compliance with enterprise guidelines, and decommissioning the device. Provisioning policies are applicable to enterprise-issued mobile devices only. Monitoring policies have traditionally been concealed so that employees are not quite aware of how, why, and when their actions are being monitored, what exactly is being monitored, and what policies govern the use of the captured data. Monitoring policies are applicable towards all mobile devices, even though there are more tools that can be used for monitoring company-owned mobile devices. (Campagna et al. 2011)

Password policies define best practices for password usage in corporate or personal environments, regardless of what type of device or application is being protected. The lack of a password is a huge security issue, and making sure that everyone who accesses the corporate data with a smartphone has a password

that meets the corporate password policy guidelines set on the device is critical. (Lim et al. 2013)

Application policies establish what mobile applications users are permitted to use while accessing the enterprise network. This policy applies mostly to company-owned mobile devices. The company has no control over the applications installed on employee-owned devices, but the employer can still impose restrictions on applications that connect to the corporate network and rely on the monitoring policies to ensure that enterprise compliance is being maintained. (Campagna et al. 2011)

## **4.2 Features**

All MDM frameworks provide the same set of core functionalities and features, which are then complemented by additional vendor-specific functionalities and capabilities (Bergman et al. 2013). MDM features include device provisioning, policy management, data encryption, backup and restore, and many more (Speed et al. 2013).

### **4.2.1 Controlling applications**

After a device has been provisioned, and the appropriate policy settings and configurations have been delivered, applications can be deployed to the device. The most popular way that consumers acquire applications for their smartphones is through application stores like the Google Play Store. This is a risky, but potentially useful distribution mechanism for the enterprise, as the employee could accidentally download malicious applications to the device. (Campagna et al. 2011)

The ability to blacklist and remove applications is an important part of ensuring that there are no unauthorized applications on employee devices. MDM solutions often allow the blacklisting of unwanted applications, which prevents users from installing these applications on their mobile devices. To be able to blacklist an application, it must have first been discovered and deemed as a threat. Another,

maybe a more secure way to control applications on the device and ensure they are safe is to whitelist applications. This basically grants the user access only to specific authorized applications. Whitelisting is a protective security method that denies all mobile apps that have not been reviewed for security in the organization. This way, new malicious applications cannot be added to the device before they have been evaluated. Maintaining and updating both whitelist and blacklist requires considerable corporate resources. (Campagna et al. 2011; Zahadat et al. 2015)

#### **4.2.2 Protecting the device from loss and theft**

It is impossible to defend against physical access to the device for long, and there is a limited amount of options regarding the countermeasures. Once an employee reports their device lost or stolen, the organization will need to take immediate action to protect data that may reside on the device. It may be difficult to differentiate accidental loss from malicious theft, which is why in uncertain situations it makes sense to be on the safe side and assume that a device has been stolen or found by someone who has malicious intent. (Dunham 2009; Bergman et al. 2013; Zahadat et al. 2015)

The primary defence against device loss or theft is to encrypt all sensitive information on the device. Devices that store sensitive data should also be protected by a passcode. MDM solutions make it possible to set a requirement for the length and strength of the passcode, and a limit for the number of incorrect login attempts before the device gets locked. This makes it much more difficult for an unauthorized person to guess the passcode and access the data on the device. Another MDM feature that protects the device from physical attacks is the ability to remotely track the location of a device via GPS. (Dunham 2009; Bergman et al. 2013)

Remote lock and remote wipe are also widely used features in MDM solutions. These actions are invoked by MDM administrators in device loss, device breach, and device noncompliance scenarios to ensure the security and integrity of the device and associated data. The device can be remotely locked so that nobody

can log in to it and access data on it. With the remote wipe function, it is possible to wipe the device clean of either all or specific data. Wiping the entire device protects the organizational data on it as well as the employee's personal data. Wiping only the areas on the device where organizational data are stored will leave employee's data on the device, but does not guarantee that some organizational data does not remain on other areas of the device after the wipe. Remote wipe is recommended to be used as a last resort in cases where the device is lost or stolen. (Dunham 2009; Bergman et al. 2013; Zahadat et al. 2015)

Each of these actions reduces the risks of losing sensitive data on lost or stolen mobile devices. In some cases, it is possible for the employees to take immediate action themselves, instead of calling the company help desk to report the incident and have them make necessary moves. In any case, it is important to react to these events as soon as possible, to decrease the risk of sensitive corporate data getting stolen from the missing device. (Campagna et al. 2011)

## **5 INTEGRATION PROCESS**

This section describes the process of creating a prototype of M-Files for Android mobile application that has been integrated with Workspace ONE mobile device management solution. Android Studio was used as the development environment, and the application was written in Java. VMware's official integration guides were followed throughout this process.

### **5.1 Workspace ONE**

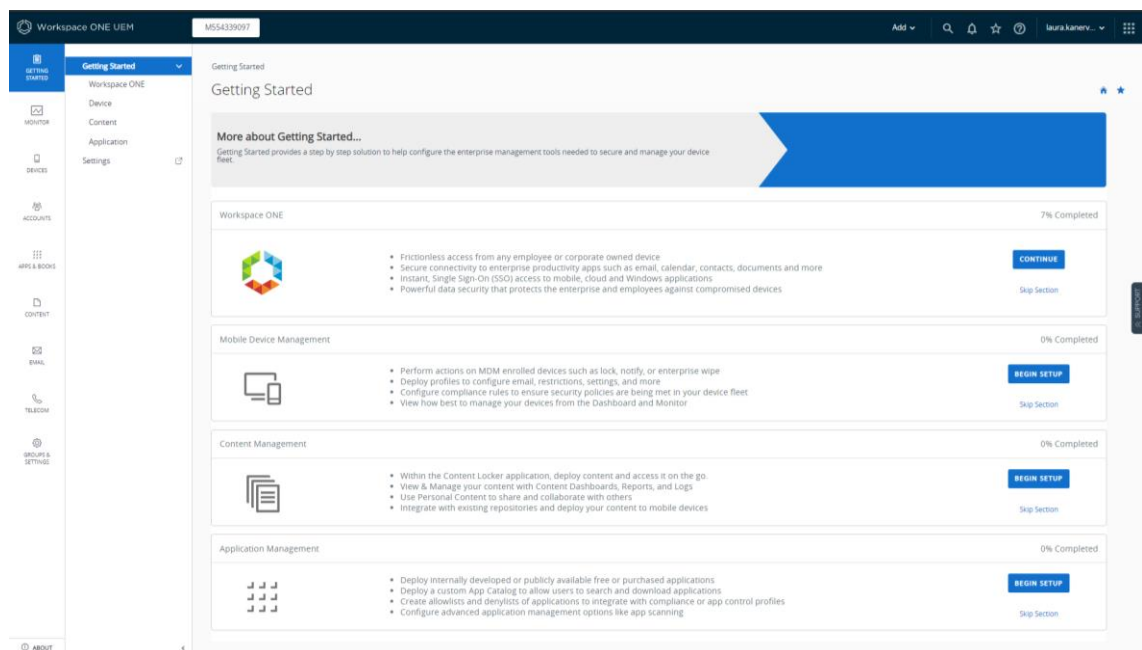
VMware Workspace ONE is a digital platform that facilitates simple and secure management of any app on any device. With Workspace ONE, access control, application management and multi-platform endpoint management meet in a single platform. Workspace ONE offers flexible management for BYOD programs and device-aware access policies amongst other capabilities. (VMware Workspace ONE n.d.)

There are multiple technical approaches to choose from when integrating Android applications with Workspace ONE. Most commonly, developers use a combination of technical approaches to meet the desired outcome. Only the software development kit (SDK) approach was used in this project. The Workspace ONE SDK, formerly known as AirWatch SDK, for Android devices can be used to enable additional capabilities that may not be available natively. The SDK is mostly used in cases that require deeper integration and functionality that for example app wrapping cannot provide. The Workspace ONE SDK for Android also works well in deployment scenarios where it is not possible to install an MDM profile on the device. The SDK version used in this integration process was AirWatch SDK 21.3. (Workspace ONE SDK for Android n.d.; Workspace ONE Technical Approaches n.d.)

#### **5.1.1 Device deployment**

MDM can be deployed through an enrolment or provisioning process. The process delivers the necessary configurations and policy settings to mobile devices and provides access to resources controlled by the MDM server. (Bergman et al. 2013)

Access to Workspace ONE management console was first needed in order to start the initial provisioning process. The management console is often referred to as the Unified Endpoint Manager (UEM). A free 30-day trial was used in making the prototype. A *My VMware* account was created, registered, and activated, and Workspace ONE environment information and credentials were received via email. The Workspace ONE UEM (Picture 2) could then be accessed.



PICTURE 2. The Workspace ONE management console.

The UEM needed to be registered as the Enterprise Mobility Management (EMM) provider with Google. Changes to the enrolment settings were also made to enable the management of Android devices (Picture 3).


Enrollment 

Authentication   **Management Mode**   Terms of Use   Grouping   Restrictions   Optional Prompt   Customization

---

Current Setting    Inherit    Override

---

 Devices enrolled through Intelligent Hub will be MDM managed by default. Enable and select the appropriate Smart Groups below to allow devices to enroll with following criteria when utilizing smart groups: OS Version, Ownership Type, and User Group. Use Adaptive Management app policies to control device management.

---

iOS	<input type="checkbox"/> ENABLED <input checked="" type="checkbox"/> DISABLED
Android	<input checked="" type="checkbox"/> ENABLED <input type="checkbox"/> DISABLED
All Android devices in this Organization Group	<input checked="" type="checkbox"/> ENABLED <input type="checkbox"/> DISABLED
Windows	<input type="checkbox"/> ENABLED <input checked="" type="checkbox"/> DISABLED

PICTURE 3. Enrolment settings to enable the management of Android devices.

A new end user account was created on the UEM, and enrolment credentials were sent to the email assigned to their account (Picture 4).

- **Enroll your device.**

Authentication may be required. Your unique credentials are below:

Server URL: [ds1678.awmdm.com](https://ds1678.awmdm.com)

Group ID: M7517125

Username: laura.kanerva

Password: Use Password Reset Link

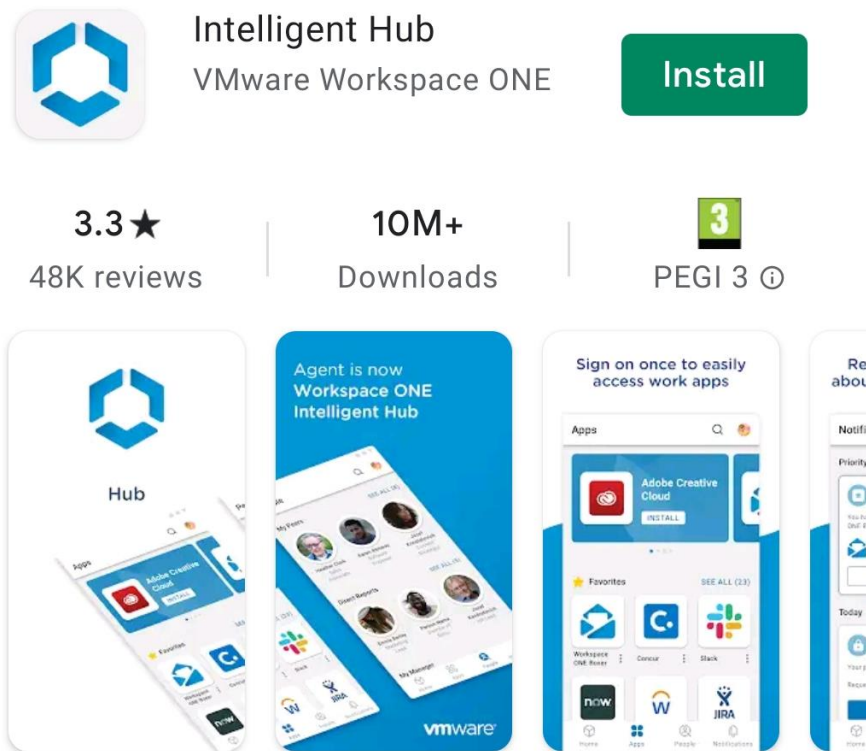
Alternatively, scan the QR code to begin enrollment:



PICTURE 4. Enrolment credentials user receives in email.

A physical Android device was needed for development purposes. Android Emulator, a virtual mobile device emulator, could normally be used to test Android applications without an actual mobile device. The emulator could not be used in

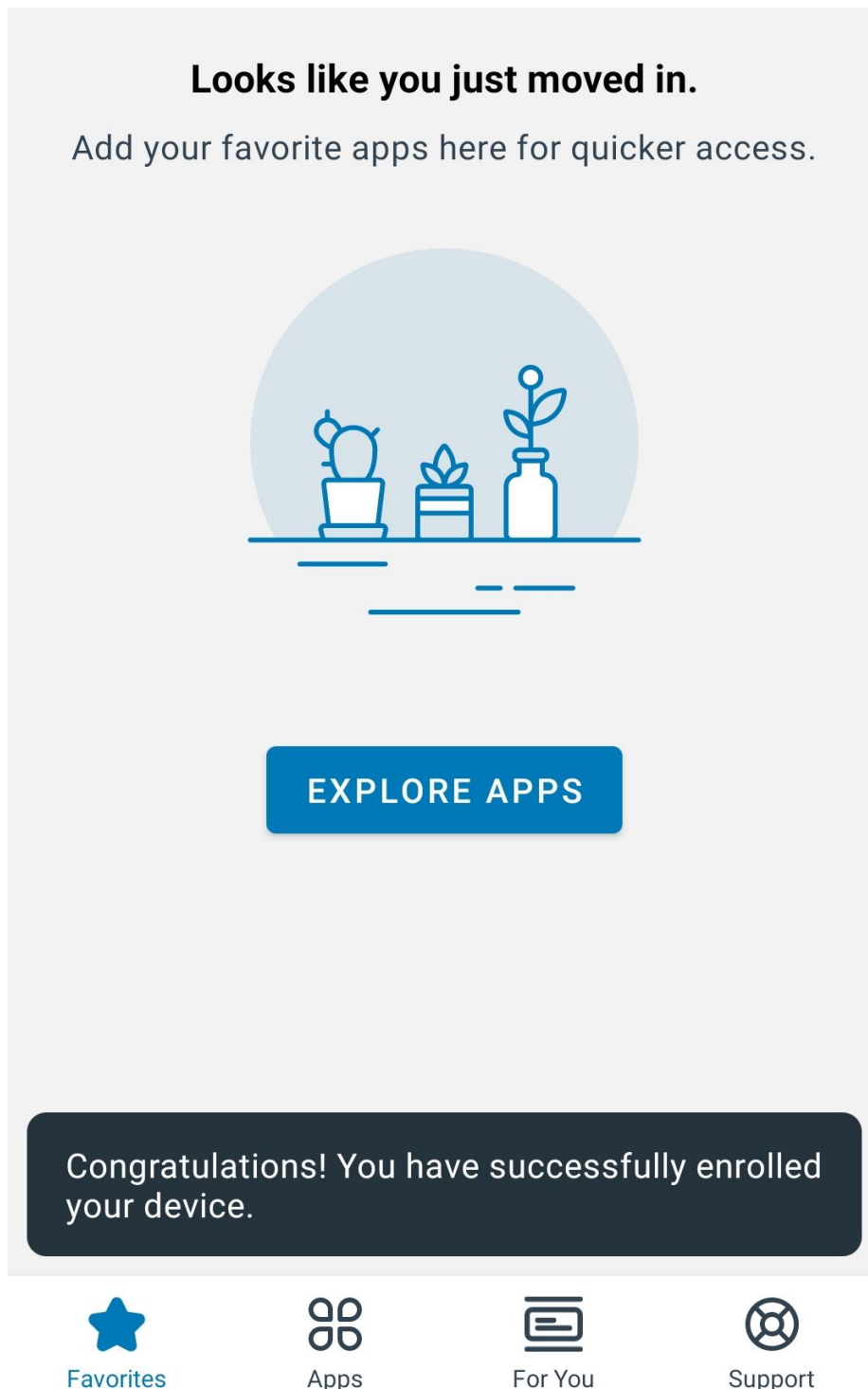
this case, because emulated devices may appear as rooted or otherwise compromised. A Nokia 8 device was primarily used instead. The device was set up for developer use, and the Workspace ONE Intelligent Hub application (Picture 5) was installed on the phone. The application is used to connect the device to the management server so that it can be provisioned.



PICTURE 5. Workspace ONE Intelligent Hub application in the Google Play store.

The device was then enrolled with the enrolment details received in email. Enrolment could have been done by either scanning the given QR code or manually entering the URL of the management server. Username and password were then asked to verify the identity of the user, and the registration process was complete (Picture 6).





PICTURE 6. Workspace ONE Intelligent Hub application after successful enrollment.

During the registration, the system verifies which policies and configurations to apply to the device based on who the user is and what type of device they use. The configuration information is sent to the mobile device, and the device is updated with new settings. (Campagna et al. 2011)

## 5.1.2 Configurations

A number of configurations and policies can be pushed down to groups of devices from the management console. Only a handful of them were tried out in this integration process.

### Chrome Browser Settings

#### Content Settings

Allow Cookies	<input type="text" value="Allow all sites to set local data"/>
Allow Cookies On These Sites	<input type="text" value="Allow all sites to set local data"/>
Block Cookies On These Sites	<input type="text" value="Do not allow any site to set local data"/>
Allow Session Only Cookies On These Sites	<input type="text" value="Keep cookies for the duration of the session."/>
Allow Images	<input type="text" value="Ex: http://www.example.com"/>
Allow Images On These Sites	<input type="text" value="Ex: http://www.example.com"/>
Block Images On These Sites	<input type="text" value="Ex: http://www.example.com"/>

PICTURE 7. Creating a device profile with Chrome browser settings.

Device profiles can be used for example to apply browser settings (Picture 7), passcode settings, network and application restrictions, and system update settings. Compliance policy rules could also be created for different device groups, regarding the device's data usage, compromised status, or OS version, for example. Actions could then be set to be performed if the device would at some point not be compliant with the rules (Picture 8). An email could be sent to the user, managed applications could be removed from the device, or a whole enterprise wipe could be performed on the device.

## Add Compliance Policy

1 Rules 2 Actions 3 Assignment 4 Summary

Match  Of The Following Rules

<input type="text" value="Interactive Certificate Profile Expiry"/>	<input type="text" value="Within"/>	<input type="text" value="0"/> Days
<input type="text" value="Last Compromised Scan"/>	<input type="text" value="Not Within"/>	<input type="text" value="4"/> Hours
<input type="text" value="MDM Terms of Use Acceptance"/>	<input type="text" value="Not Within"/>	<input type="text" value="4"/> Hours
<input type="text" value="Model"/>	<input type="text" value="Is Not"/>	<input type="text" value="Android - Android"/>
<input type="text" value="OS Version"/>	<input type="text" value="Is"/>	<input type="text" value="Android - Android 2.2"/>

Is  
 Is Not  
 Greater than  
 Greater than or equal to  
 Less than or equal to

PICTURE 8. Examples of compliance policy rules.

App groups could be created for allowed, denied, and required applications (Picture 9). These settings could be pushed to devices based on their model, operating system, ownership status, or organizational group.

Type*	Platform*	Name*
<input type="text" value="Denylist"/>	<input type="text" value="Android"/>	<input type="text" value="Blacklisted Apps"/>

Application Name*	Application ID*
<input type="text" value="Snapchat"/> <input type="button" value="🔍"/>	<input type="text" value="com.snapchat.android"/>
<input type="text" value="Facebook"/> <input type="button" value="🔍"/>	<input type="text" value="com.facebook.katana"/>
<input type="text" value="Instagram"/> <input type="button" value="🔍"/>	<input type="text" value="com.instagram.android"/>

PICTURE 9. Blacklisting applications in the Workspace ONE UEM.

An SDK profile could be made with authentication settings (Picture 10), restrictions, and many more options.

## Authentication

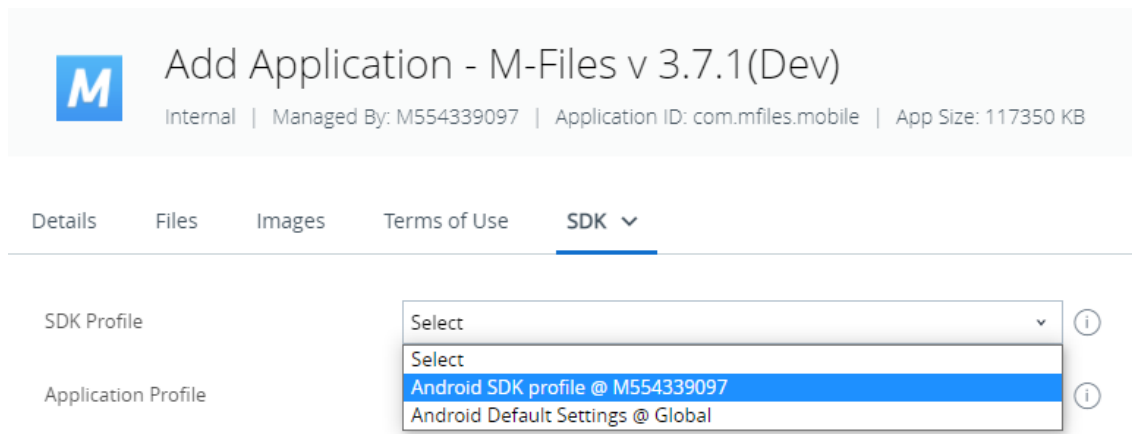
Enable Kerberos	<input type="checkbox"/>
Single Sign-On	<input type="checkbox"/>
Authentication Type	Username and Password <span>▼</span>
Biometric Mode	Disabled <span>▼</span> ⓘ
Authentication Timeout	1 <input type="text"/> minute(s) <span>▼</span> ⓘ
Maximum Number Of Failed Attempts	4 <span>▼</span>
Integrated Authentication	<input type="checkbox"/>

PICTURE 10. Authentication settings on an SDK profile.

### 5.1.3 Distributing applications

Applications could be delivered to the device in several ways, including direct download from an application store or over-the-air delivery. The advantage of distributing applications to devices with MDM, in comparison to having the employee find the appropriate application on their own, is that it almost certainly ensures that the employee selects the right application, and not a malicious counterfeit version of a legitimate application. (Campagna et al. 2011)

Both public and internal applications can be managed with Workspace ONE. Public applications are available within public app stores, whereas internal applications are typically company-specific. Internal Android applications are uploaded to the management console via an Android application package (APK) file. A signed APK file needed to be generated in Android Studio for this purpose. When uploading an application to the management console, a previously created SDK profile could be assigned to it (Picture 11). This way, the application works according to the profile settings.



PICTURE 11. Assigning an SDK profile to an internal application.

The application could then be assigned to a group of devices and delivered at a set time either automatically or on demand, when the user chooses to download it, as shown in picture 12.

### Distribution

Name \* M-Files Assignment

Description

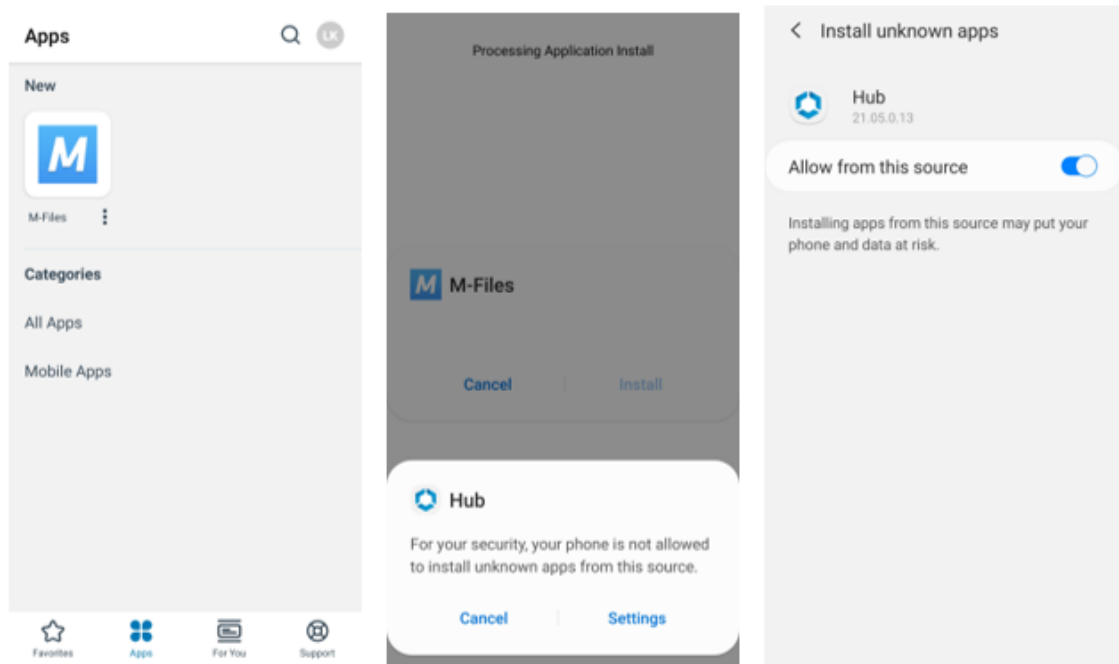
Assignment Groups \*

Deployment Begins \* 06/27/2021  12:00 AM (GMT-05:00) Eastern Time (US & Canada)

App Delivery Method \*  Auto  On Demand

PICTURE 12. Selecting the distribution group, time and method for an application.

After the application has been deployed, it can be installed on the device from the Intelligent Hub. Rights to install apps to the device needed to be granted (Picture 13).



PICTURE 13. Installing the application from the Hub.

## 5.2 Integrating Workspace ONE into Android application

### 5.2.1 Integration preparation

The application that was being integrated needed to be installed from the Hub at least once during the integration work. The first installation was of a non-integrated version of the application. The subsequent installations of integrated versions of the application were made using the Android Debug Bridge (adb) tool, that enables the computer to communicate with a mobile device via a USB cable. The adb installations were upgrades, and the application was never uninstalled after the first installation via Workspace ONE.

Installing a non-integrated version of the application first is smart, because the whole installation process via Workspace ONE involves tasks that many are unfamiliar with. It is better to first get to know the management console and set up signed builds with the application in a familiar working state. (Hawkins 2020)

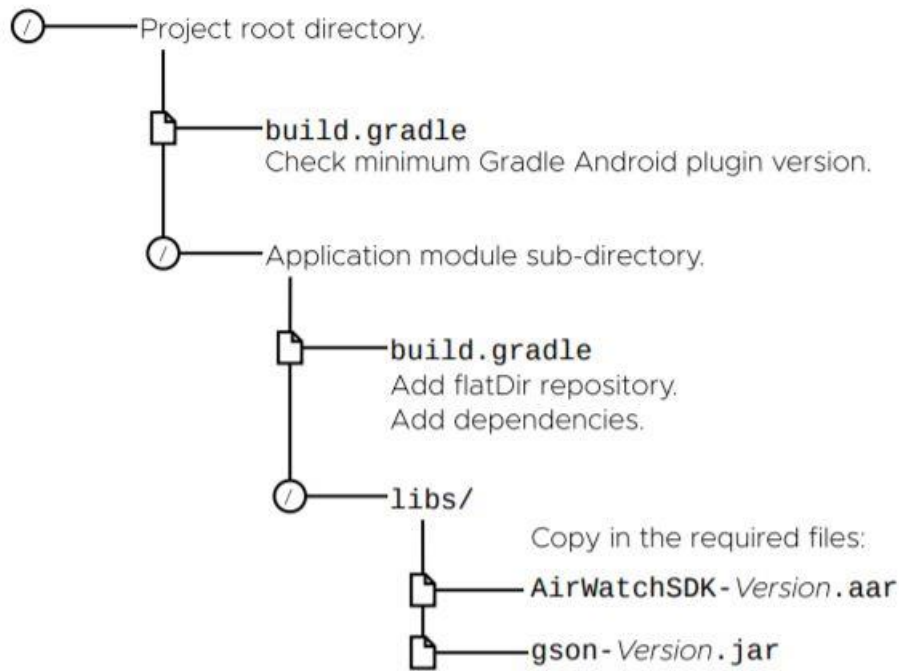
The Workspace ONE SDK for Android was downloaded from the *My Workspace ONE* portal. The Workspace ONE SDK can be integrated to different levels, depending on requirements. Client level integration requires the least changes to the application source code, but it also only enables features such as enrolment status, user information, and device compromise detection. (Kronemeyer & Hawkins 2021)

Framework level integration requires more work by the application developer. In addition to the Client level features, Framework integration enables features such as full SDK profile, Single Sign On (SSO) for apps, end user authentication, inactivity lock, application data encryption, and copy paste restrictions. Full Framework integration also requires modifications to the app user interface. (Kronemeyer & Hawkins 2021)

There is a third level called Networking above Framework, that makes available all Client and Framework integration features, tunneling of application data, and certificate-based authentication, for example. Branding could also be applied through the integration. The prototype created during this project did not utilize these levels of integration.

### **5.2.2 Framework level integration**

The actual integration started with adding the Client SDK. The first step was to set up the build configuration and files. Picture 14 illustrates the project directory structure, and the locations where following changes were made.



PICTURE 14. Project structure and locations of changes. (Workspace ONE Base Integration Guide for Android 2021)

In the project build.gradle file, the Gradle Android plugin version was updated to the minimum requirement to ensure compatibility. In the application build.gradle file, references to the required libraries were added inside the dependencies block.

```

139 dependencies {
140     // ...
141
142     // Airwatch Client SDK
143     // Workspace ONE libraries that are part of the SDK.
144     implementation (name:'AirWatchSDK-21.3', ext:'aar')
145     implementation(name:"ws1-android-logger-1.2.0", ext:'aar')
146     implementation(name:"FeatureModule-android-2.0.0", ext:'aar')
147     implementation(name:"sdk-fm-extension-android-1.1", ext:'aar')
148     // Third party libraries that are hosted remotely.
149     implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.3.3'
150     implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.3.3'
151     implementation "androidx.lifecycle:lifecycle-runtime:2.3.1"
152     implementation "androidx.lifecycle:lifecycle-extensions:2.2.0"
153     annotationProcessor ("androidx.lifecycle:lifecycle-compiler:2.3.1") {
154         exclude group:'com.google.guava', module:'guava'
155     }
156     implementation ("org.jetbrains.kotlinx:kotlinx-serialization-runtime:0.20.0")
157

```

PICTURE 15. References added to integrate Workspace ONE at the Client level.



Picture 15 indicates the required libraries. Library files that are either part of the SDK or distributed with the SDK were copied from the downloaded SDK distribution into the project, under the application module sub-directory, in the libs sub-directory (Picture 14).

The next step was to implement an AirWatch SDK Service class. An implementation of a specific Android broadcast receiver and service was added to the application to support various essential notifications that the Client SDK receives from the management console. A new class was implemented as shown in picture 16.

```
11 |  
12 |  
13 |  
14 | public class AirWatchSDKIntentService extends AirWatchSDKBaseIntentService {  
15 |     @Override  
16 |     protected void onApplicationConfigurationChange(Bundle bundle) {  
17 |  
18 |     }  
19 |  
20 |     @Override  
21 |     protected void onApplicationProfileReceived(Context context, String s, ApplicationProfile applicationProfile) {  
22 |  
23 |     }  
24 |  
25 |     @Override  
26 |     protected void onClearAppDataCommandReceived(Context context, ClearReasonCode clearReasonCode) {  
27 |  
28 |     }  
29 |  
30 |     @Override  
31 |     protected void onAnchorAppStatusReceived(Context context, AnchorAppStatus anchorAppStatus) {  
32 |  
33 |     }  
34 |  
35 |     @Override  
36 |     protected void onAnchorAppUpgrade(Context context, boolean b) {  
37 |  
38 |     }  
39 | }  
40 |
```

PICTURE 16. Implementation of the AirWatchSDKIntentService class.

In the Android manifest file (Picture 17), permission, receiver and service declarations were added. These were all the steps required to continue to the Framework-level integration. The application was built to confirm no mistakes had been made.

```

13     <uses-permission android:name="com.airwatch.sdk.BROADCAST" />
14
15     // ...
16
17     <application>
18
19         // ...
20
21         <!-- AirWatch -->
22         <receiver
23             android:name="com.airwatch.sdk.AirWatchSDKBroadcastReceiver"
24             android:permission="com.airwatch.sdk.BROADCAST" >
25             <intent-filter>
26                 <action android:name=".airwatchsdk.BROADCAST" />
27             </intent-filter>
28             <intent-filter>
29                 <action
30                     android:name="com.airwatch.intent.action.APPLICATION_CONFIGURATION_CHANGED"
31                     />
32                 <data android:scheme="app" android:host="com.mfiles.mobile" />
33             </intent-filter>
34             <intent-filter>
35                 <action android:name="android.intent.action.PACKAGE_ADDED" />
36                 <action android:name="android.intent.action.PACKAGE_REMOVED" />
37                 <action android:name="android.intent.action.PACKAGE_REPLACED" />
38                 <action android:name="android.intent.action.PACKAGE_CHANGED" />
39                 <action android:name="android.intent.action.PACKAGE_RESTARTED" />
40                 <data android:scheme="package" />
41             </intent-filter>
42         </receiver>
43         <service android:name=".service.AirWatchSDKIntentService"
44             android:permission="android.permission.BIND_JOB_SERVICE"/>
45
46     </application>

```

PICTURE 17. The Android manifest file.

The Framework level required more library references to be added to the application build.gradle file (Picture 18). The M-Files Android application already required different versions of some of the same libraries required by the SDK. These libraries were updated in the process, since the SDK is only supported with versions it requires. Compatibility issues were encountered with other versions, especially older ones. The required library files were again copied to the same location as previously. Enabling Kotlin support was needed at this point, even if the source code is written in Java.

```

139 dependencies {
140     // ...
141
142     // Following lines are added to integrate Workspace ONE at the Framework level
143     def room_version = "2.2.4"
144     // Workspace ONE libraries that are part of the SDK.
145     implementation(name:'ws1-sdk-oauth-api-lib-1.2.0', ext:'aar')
146     implementation(name:'SCEPClient-1.0.16', ext:'aar')
147     implementation(name:'AWComplianceLibrary-2.3.6', ext:'aar')
148     implementation(name:'AWFramework-21.3', ext:'aar')
149     implementation(name:'VisionUx-1.5.0.a', ext:'aar')
150     implementation(name:'CredentialsExt-102.1.0', ext:'aar')
151     implementation(name:"chameleon-android-1.1.1.8--20201116T093924Z", ext:'aar')
152     implementation(name:"module-settings-1.2.0.1--20201125T150536Z", ext:'aar')
153     implementation(name:"settings-1.3.1.7--20201201T114153Z", ext:'aar')
154     implementation(name:"opdata-android-1.5.0.4--20201201T152231Z", ext:'aar')
155     implementation(name:"attributesprovider-1.3.1.7--20201201T114153Z", ext:'aar')
156     implementation(name:"encryptedpreferencesprovider-1.3.1.7--20201201T114153Z", ext:'aar')
157     implementation(name:"httpprovider-1.3.1.7--20201201T114153Z", ext:'aar')
158     implementation(name:"memoryprovider-1.3.1.7--20201201T114153Z", ext:'aar')
159     implementation(name:"supercollider-1.0.7-ndk-r21c", ext:'aar')
160     implementation(name:"work-hour-access-sdk-android-1.0.0", ext:'aar')
161     // Third party libraries that are distributed with the SDK.
162     implementation("com.squareup.moshi:moshi-kotlin:1.8.0"){
163         exclude group: 'com.squareup.okio', module: 'okio'
164         exclude group: 'com.squareup.moshi', module: 'moshi'
165     }
166     kapt "com.squareup.moshi:moshi-kotlin-codegen:1.8.0"
167     implementation 'com.squareup.moshi:moshi:1.8.0'
168     implementation 'com.squareup.moshi:moshi-adapters:1.8.0'
169     implementation 'com.squareup.okio:okio:1.17.2'
170     implementation 'com.google.zxing:core:3.3.3'
171     implementation 'com.google.code.gson:gson:2.4'
172     // Third party libraries that are hosted remotely.
173     implementation 'androidx.security:security-crypto:1.0.0-rc02'
174     kapt "androidx.lifecycle:lifecycle-compiler:2.2.0"
175     implementation 'com.google.android.gms:play-services-safetynet:17.0.0'
176     implementation 'androidx.legacy:legacy-support-v13:1.0.0'
177     implementation 'androidx.appcompat:appcompat:1.1.0'
178     implementation 'androidx.cardview:cardview:1.0.0'
179     implementation 'com.google.android.material:material:1.1.0'

```

PICTURE 18. References added to integrate Workspace ONE at the Framework level.

Next step in the integration process was the Framework initialization. Instead of having to create a new subclass, the M-Files application already had an existing Android Application subclass that was chosen as the Framework initialization class. The subclass was updated with new implementation declaration, adding an AWSDKApplicationDelegate instance as a property, overriding methods, and implementing other AWSDKApplication methods by calling the same method in the AWSDKApplicationDelegate instance (Picture 19).

```

60     public class App extends WorkApplication implements AWSDKApplication {
61
62         // SDK Delegate.
63         private final AWSDKApplicationDelegate awDelegate = new AWSDKApplicationDelegate();
64
65         // ...
66
67         @Override
68         public Object getSystemService(String name) {
69             return this.getAWSystemService(name, super.getSystemService(name));
70         }
71
72         // ...
73
74         @Override
75         public void initializeLogger() { awDelegate.initializeLogger(); }
76
77
78         @Override
79         public boolean isAppProcess() { return awDelegate.isAppProcess(); }
80
81
82
83

```

PICTURE 19. Preview of the code changes made in the Framework initialization phase of the integration process.

Finally, the application was built and run to confirm that no mistakes had been made. A few issues were encountered during the integration process and testing.

### 5.2.3 Testing and troubleshooting

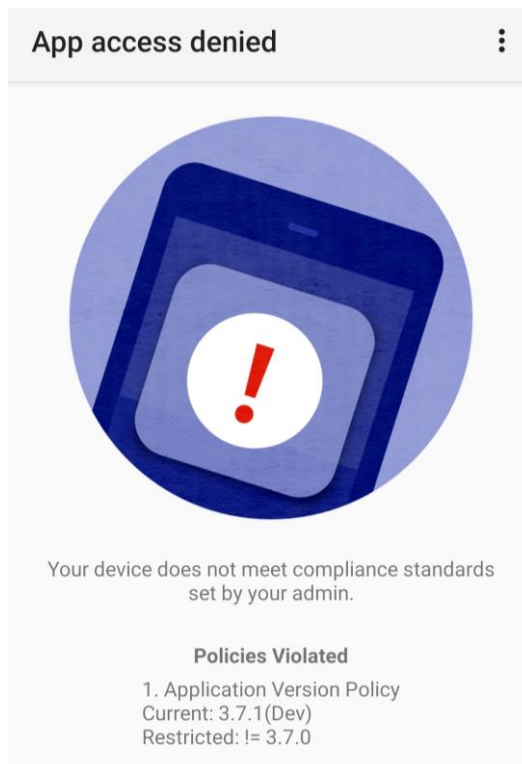
Testing and recording the MDM controls and seeing how they are enforced on different devices is essential (Lim et al. 2013). The integration was tested on two smartphones, Nokia 8, that could be considered an employee-owned device, and Samsung Galaxy S10e, that was a company-owned device. Developer options and USB debugging were enabled on the test devices. The devices were connected to a computer via USB one at a time. The application was run on the connected device.

Many build errors happened because of duplicate and incompatible dependencies. The duplicates could easily be removed, but working around compatibility issues was difficult in an application of this size. Upgrading some dependencies led to certain application features failing, since they were not supported in the later versions of the libraries in question. Fixing these crashing features takes an indeterminate amount of time, but is evident.

A `ClassNotFoundException` was also encountered, because a reasonably new Android class was used in the application, and the mobile phone used for testing was running an older version of Android that did not support the class in question. This was surprising, since the device was selected based on the minimum requirements stated in the official Workspace ONE documentation. The device had to be retired from testing.

There were many more issues with the support and documentation regarding Workspace ONE SDK integration. At the time of creating the prototype, the documentation was outdated, and there was very little support available online. Most of the official documents have since been updated, but still lack some extremely important details that had to be figured out the hard way during the integration process. The Workspace ONE UEM was also deemed difficult to use without much up-to-date instructions available, which is why many features could not be properly investigated.

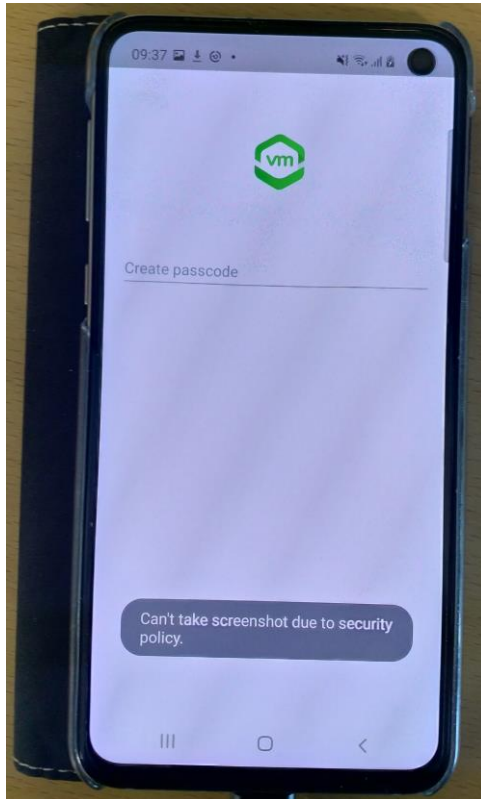
Regardless of the many issues encountered, some features were successfully tested. For example, access to managed applications distributed through the MDM could be denied, if the device violated set compliance policies (Picture 20). The passcode policy was also effectively enforced (Picture 21), as well as the security policy restricting the use of screenshots (Picture 22). A version of device wipe was also tried out (Picture 23).



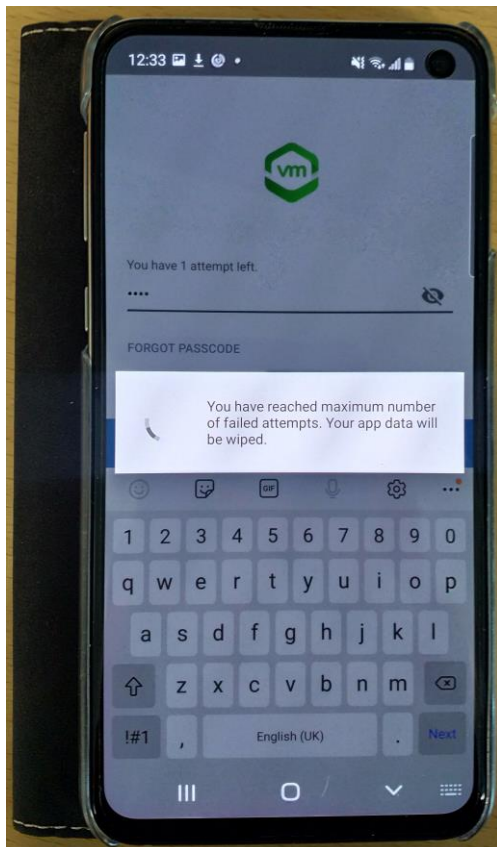
PICTURE 20. Access to managed application was denied because an incorrect version number.

The screenshot displays a 'Single Sign On' screen with a blue hexagonal logo at the top. Below the logo, the text 'Single Sign On' is centered. There are two input fields: 'New Passcode' and 'Confirm Passcode', both with blue borders. A grey 'SUBMIT' button is positioned to the right of the 'Confirm Passcode' field. At the bottom, a paragraph of text provides passcode requirements: 'Your passcode should be at least 4 character(s) long have at least a number and an alphabet and 1 symbol(s) not have repeating / ascending /descending character(s)'.

PICTURE 21. The application requires the creation of a passcode with certain conditions.



PICTURE 22. A security policy restricting the ability to take screenshots.



PICTURE 23. App data getting wiped after reaching the maximum number of failed login attempts set in the configurations.

## 6 DISCUSSION

The integration process was overall successful, although challenging. Sample code and documentation was available, but not always up to date, and there was very little support available online. A lot of work would be required to extend classes, replace components, make the app compatible with newer dependencies, and overall fully integrate the application with Workspace ONE. Of course, as M-Files' clients requesting this MDM solution support would make all the configurations in the Workspace ONE management console themselves, the development becomes slightly easier. Based on the process of creating this prototype, the time estimation for full integration would be 4 to 8 weeks, depending on the number of developers working on it.

Mobile device management is not a definite solution to the mobile security problem. It works well for facile vulnerabilities, but it is still possible to bypass the security measures it enforces. MDM should not be expected to miraculously fix all the problems, but only to alleviate some of the symptoms. It may take time to get used to using an MDM solution, and it may be expensive. Regardless, it is worth it to have a tool in use that helps keep up with the rapid changes of the mobile landscape. MDM and related technologies can notably contribute to the overall mobile security posture if designed and deployed thoughtfully. (Bergman et al. 2013; Lim et al. 2013)

As useful and important as it is to implement strict controls regarding passwords, network connections, and other possibly threatening features of mobile devices, regularly educating the users about the importance of security and proper device management takes priority. Providing security awareness training and detailed information about the threats against them effectively alleviates the users' behavioral security issues. The more they know, the better they can defend themselves. (Lim et al. 2013; Weichbroth & Łysik 2020)

Mobile application developers can also contribute to the mobile security by creating more secure applications. Mobile development is often outsourced and moves fast due to the lack of resources or to the urge to keep up with clients'



expectations. Security could be an afterthought in these situations. It is essential for application developers to also be educated about the risks and understand what it is that they are protecting by writing secure code. (Bergman et al. 2013)

With the growing use of mobile devices to access and store important personal and company information, there is a clear need for more research that focuses on the security behavior of mobile device users. Users might often overestimate their own abilities and believe that the consequences of threats to their personal devices are not as severe as they are. This links to the actions users take, and should be researched more. It would also be useful for future research to investigate how other factors impact information security behavior. (Thompson et al. 2017)

*Bring your own device* is a significant part of recent history, and will be a part of the future as well. It can be predicted that more and more malware and attacks against mobile devices will be seen, which is why making every possible effort towards enhancing mobile security is indispensable.

## REFERENCES

Bergman, N., Stanfield, M., Rouse, J. & Scambray, J. 2013. Hacking Exposed Mobile: Security Secrets & Solutions. New York: McGraw-Hill Education.

Campagna, R., Iyer, S. & Krishnan, A. 2011. Mobile Device Security for Dummies. Indianapolis: Wiley Publishing, Inc.

Chadd, K. 2018. Cybercriminals: Who are they and what do they do? Avast. Published on 19.9.2018. Read on 25.10.2021. <https://blog.avast.com/who-and-what-are-cybercriminals-avast>.

Cyber organized crime activities. E4J University Module Series: Cybercrime. Module 13: Cyber Organized Crime. UNODC. Published in 2019. Read on 31.10.2021. <https://www.unodc.org/e4j/en/cybercrime/module-13/key-issues/cyber-organized-crime-activities.html>.

Dunham, K. 2009. Mobile Malware Attacks and Defence. Burlington: Elsevier, Inc.

Gonzalez, D. 2015. Managing Online Risk: Apps, Mobile, and Social Media Security. Oxford: Butterworth-Heinemann.

Hawkins, J. 2020. Integration Overview. Workspace ONE for Android. Read on 16.4.2021. [https://github.com/vmware-samples/workspace-ONE-SDK-integration-samples/blob/main/IntegrationGuideForAndroid/Guides/01Overview/WorkspaceONE\\_Android\\_IntegrationOverview.md](https://github.com/vmware-samples/workspace-ONE-SDK-integration-samples/blob/main/IntegrationGuideForAndroid/Guides/01Overview/WorkspaceONE_Android_IntegrationOverview.md).

Kronemeyer, B. & Hawkins, J. 2021. Integration Preparation Guide. Workspace ONE for Android. Read on 16.4.2021. [https://github.com/vmware-samples/workspace-ONE-SDK-integration-samples/blob/main/IntegrationGuideForAndroid/Guides/02Preparation/WorkspaceONE\\_Android\\_IntegrationPreparation.md](https://github.com/vmware-samples/workspace-ONE-SDK-integration-samples/blob/main/IntegrationGuideForAndroid/Guides/02Preparation/WorkspaceONE_Android_IntegrationPreparation.md).

Lim, I., Coolidge, E. & Hourani, P. 2013. Securing Cloud and Mobility: A Practitioner's Guide. Boca Raton: Auerbach Publications.

Mobile Operating System Market Share Worldwide. N.d. GlobalStats statcounter. Read on 13.11.2021. <https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-202010-202110-bar>.

Mobile operating systems. N.d. Uswitch Mobiles. Read on 19.11.2021. <https://www.uswitch.com/mobiles/guides/mobile-operating-systems/>.

Poetker, B. 2021. The Mobile Operating Systems That Matter Right Now. G2. Published on 21.5.2021. Read on 19.11.2021. <https://www.g2.com/articles/mobile-operating-systems>.

Sarangam, A. 2021. 10 Types Of Hackers To Be Aware Of In 2021. Jigsaw Academy. Published on 8.3.2021. Read on 1.11.2021. <https://www.jigsawacademy.com/blogs/cyber-security/different-types-of-hackers>.

Speed, T., Nykamp, D., Anderson, J. & Nampalli, J. 2013. *Mobile Security: How to Secure, Privatize, and Recover Your Devices*. Birmingham: Packt Publishing Ltd.

Steinz, A. & Pulkkinen, V. 2021. Secure Development training. M-Files internal training on 16.6.2021. Tampere.

Thompson, N., McGill, T. & Wang, X. 2017. "Security begins at home": Determinants of home computer and mobile device security behavior. *Computers & Security* 70, 376-391.

Townsend, K. 2020. *The Authentication Puzzle*. Avast. Published on 3.7.2020. Read on 6.11.2021. <https://blog.avast.com/the-importance-of-authentication-avast>.

Verton, D. 2001. Black Hat Highlights Real Danger of Script Kiddies. *Computerworld*. Published on 23.7.2001. Read on 15.11.2021. <https://www.computerworld.com/article/2581986/black-hat-highlights-real-danger-of-script-kiddies.html>.

VMware Workspace ONE. N.d. VMware. Read on 3.6.2021. <https://www.vmware.com/fi/products/workspace-one.html>.

Weichbroth, P & Łysik, Ł. 2020. *Mobile Security: Threats and Best Practices*. Mobile Information Systems, vol. 2020.

Workspace ONE Base Integration Guide for Android. N.d. VMware. Read on 2.4.2021. [https://github.com/vmware-samples/workspace-ONE-SDK-integration-samples/blob/main/IntegrationGuideForAndroid/Guides/03BaseIntegration/WorkspaceONE\\_Android\\_BaseIntegration.md](https://github.com/vmware-samples/workspace-ONE-SDK-integration-samples/blob/main/IntegrationGuideForAndroid/Guides/03BaseIntegration/WorkspaceONE_Android_BaseIntegration.md).

Workspace ONE SDK for Android. N.d. The VMware {code} community program. Read on 10.6.2021. <https://developer.vmware.com/web/sdk/Native/airwatch-android>.

Workspace ONE Technical Approaches. N.d. The VMware {code} community program. Read on 10.6.2021. <https://developer.vmware.com/web/workspace-one/technical-approaches>.

Wu, D., Moody, G., Zhang, J. & Lowry, P. 2020. Effects of the design of mobile security notifications and mobile app usability on users' security perceptions and continued use intention. *Information & Management* 57.

Zahadat, N., Blessner, P., Blackburn, T & Olson, B. 2015. BYOD security engineering: A framework and its analysis. *Computers & Security* 55, 81-99.

How to create a mobile device management policy: 9 best practices. N.d. Helixstorm. Read on 1.11.2021. <https://www.helixstorm.com/blog/how-to-create-a-mobile-device-management-policy/>.