Inactivity Wipe

Inactivity Wipe is a feature of the Workspace ONE® platform. The feature protects data at rest in a mobile application that has integrated the Workspace ONE mobile software development kit. Enterprise data will be deleted from the application when an extended period of user inactivity is detected.

The feature is controlled by configuration in the Workspace ONE Unified Endpoint Manager (UEM) console, and applied by the Workspace ONE mobile Software Development Kit (SDK). The feature is available in the SDK for Android, and in the SDK for iOS.

Table of Contents

Overview

The inactivity wipe feature can protect data in a mobile application that has integrated the Workspace ONE mobile SDK. If the feature is in use, the SDK will delete enterprise data from the application when an extended period of user inactivity is detected. The SDK can process an inactivity wipe when the application is offline or otherwise unable to receive wipe commands from the management console.

Description

The feature works as follows.

- Inactivity wipe is configured in the management console.

  The feature can be configured in any type of Workspace ONE deployment. It works on both managed and registered devices. (Registered devices were sometimes referred to as unmanaged devices.)

  The configuration of this feature can be one of the following:

  - Feature is on, and the allowed inactivity period is specified in days.
  - Feature is off.

  The feature is off by default.

- The inactivity wipe configuration is retrieved by the SDK instance in a mobile application. Retrieval attempts are made according to a defined schedule and set of events depending on device support.
- The SDK checks for inactivity if a limit is configured:
  - Whenever the application initializes the SDK.

    SDK initialization will take place, for example, when the application is launched by the user and isn't running already.

    See Initialization for Kotlin and Initialization for Swift for details of the programming interfaces.

  - Whenever the application instructs the SDK to check compliance.

    This supports scenarios in which the application runs without initializing the SDK, such as background synchronisation.

    See Compliance Check for Kotlin and Compliance Check for Swift for details of the programming interfaces.

  - Periodically when the application is in background, if supported by the device.

Some Android devices support background checking. No iOS nor iPadOS device supports background checking.

- The user is classed as active when either of the following is true:
  - The application is running in device foreground.
  - Part of the application user interface is opened, such as in a notification interaction, for Android, or in an application extension, for iOS.

  In some cases, the application code must notify the SDK when the user is active. See Inactivity Timer Reset for Kotlin and Inactivity Timer Reset for Swift for details of the programming interfaces.
- If the length of time that has passed since the last user activity is longer than the allowed inactivity period, the SDK executes an application wipe.

  An application wipe includes the following.
  - All SDK application data encryption keys are deleted.
    - In the SDK for Kotlin, this includes the keys that underlie, for example, the following classes in the programming interface:
      - `MasterKeyManager`
      - `CertificateManager`
      - `SecurePreferences`
    - In the SDK for Swift, this includes the keys that underlie the `AWController encrypt` programming interface.

    Any data encrypted by those interfaces cannot be decrypted after an inactivity wipe.
  - All credentials from the application's registration with the console are deleted, including any electronic certificates.

    If Workspace ONE Single Sign-On (SSO) for apps is in use, then the application will attempt to re-register automatically. Another app, such as the anchor app, must be available and mustn't itself have been wiped. Automatic registration will be as the same end user.

    If SSO isn't in use, or automatic registration isn't available, then the user will have to register manually.

  Note that the SDK doesn't report inactivity wipe to the UEM.

  In some cases, the SDK depends on the application code to execute the wipe. See Initialization for Kotlin and Initialization for Swift for details of the programming interfaces.

- After executing an inactivity wipe, the SDK shows an informative message in the user interface.

  In some cases, the SDK depends on the application code to show the message. See Initialization for Kotlin and Initialization for Swift for details of the programming interfaces.

  See also the User Interface for Android and User Interface for iOS screen captures in the appendix to this document.

- The SDK notifies the application when or after the inactivity wipe has been processed. See Initialization for Kotlin and Initialization for Swift for details of the programming interfaces.

Transition

Transition between this feature being on and off for an individual application on a device works as follows.

- Off to on, the sequence will be:

  1. The on setting is configured in the management console.
  2. Later, the updated configuration is retrieved by the SDK instance in the application.
  3. The next and subsequent times the application is started, the SDK will check for inactivity and take action if the check fails.

  In theory, an inactivity check could be made at the point of retrieval. In practice, retrieval can only take place when the user is active, so that check would always pass.

- On to off, the sequence will be:

  1. The off setting is configured in the management console.
  2. Next time the application starts, the SDK checks for inactivity as before.

     If the check fails, an inactivity wipe will be processed. After the inactivity wipe, the user will register manually, or be re-registered automatically.
  3. The updated configuration is retrieved by the SDK instance in the application.
  4. The next and subsequent times the application is started, the SDK won't check for inactivity.

  Note that updating the configuration in the UEM won't save applications that have already exceeded the inactivity period, nor applications that exceed the period before they can retrieve the updated configuration.

Transitions from longer to shorter inactivity periods, or shorter to longer, will be handled similarly to transitions from off to on, or on to off, respectively.

Unit of Measure

The unit of measure for inactivity configuration is days. One day is interpreted by the SDK as twenty-four hours of elapsed time. The interpretation ignores calendar days, changes in time zone, and daylight saving time.

The following example illustrates the interpretation for time zones.

Scenario:

- Inactivity period for the application is configured as 2 days.
- User is active until Monday at 6pm Pacific, then travels to the Eastern time zone.
- User remains inactive until Wednesday 7pm Eastern and then opens the application.

The user has been inactive for:

- 49 hours of calendar time, 2 days plus 1 hour.
- 46 hours of elapsed time, 7pm Eastern is 4pm Pacific.

Because the SDK interprets a period of 2 days as 48 hours elapsed, the inactivity period hasn't expired.

Integration

To integrate the feature into your application, follow the instructions below.

Configuration

This feature can be configured in the Workspace ONE management console. The following instructions are intended for application developers or other users wishing to try out the feature quickly. Full documentation can be found in the online help.

1. Log in to the management console.

   The dashboard will be displayed.

2. Select an organization group.

   By default, the Global group is selected.

3. Navigate to: Groups & Settings, All Settings, Apps, Settings and Policies, SDK App Compliance.

   This opens the SDK App Compliance configuration screen, on which a number of settings can be switched on and off, and configured.

4. For the Application Inactivity setting, select Enabled.

   When Enabled is selected, further controls will be displayed.

5. Use the controls to configure the inactivity wipe feature.

6. Select Save to commit your changes to the configuration.

See also the Console User Interface screen capture in the appendix to this document.

Programming Interface for Kotlin

The following parts of the SDK programming interface for Kotlin are relevant to this feature.

Use of this feature requires Framework integration of the SDK for Android, i.e. adding the AWFramework library.

Initialization

The following code snippets illustrate the general way to initialize the SDK, and handle inactivity wipe. Note that the SDK for Kotlin depends on the application code to execute the wipe at a time when it can do so without losing the integrity of its data.

```
class AirWatchSDKIntentService: AirWatchSDKBaseIntentService() {

    override fun onClearAppDataCommandReceived(context: Context, reasonCode: ClearReasonCode) {
        // This method is invoked by the SDK to wipe the application data, for example
        // following a failed inactivity check.
        //
        // Run application code to do the following from here:
        //
        // -   Delete any data that was encrypted with a key that is being deleted.
        // -   Delete any data that wasn't encrypted.
        // -   Delete any data that was encrypted with a key from elsewhere.

        // The following SDK method must be called at some point in the implementation.
        SDKContextManager.getSDKContext().sdkClearAction.clear(SDKClearAction.Type.ALL)

        // The application must show an informative message to the user, if the wipe is due to
        // a failed inactivity check. The SDK comes with a built-in screen for this purpose.
        if (reasonCode == ClearReasonCode.APP_INACTIVITY) {
            // The following SDK method will show the built-in screen, if the application is in
            // foreground and able to open the screen.
            const shown = ComplianceViolationActivity
            .showInactiveComplianceViolationScreen(context)
            // `shown` will be false if the screen wasn't shown. The application must record the
            // wiped state persistently and show an informative message at the next opportunity
            // in that case.
        }
    }

    override fun onApplicationConfigurationChange(
        applicationConfiguration: Bundle
    ) {
        // ...
    }

    override fun onApplicationProfileReceived(
        context: Context,
        profileId: String,
        appProfile: ApplicationProfile
    ){
        // ...
    }

    override fun onAnchorAppStatusReceived(context: Context, appStatus: AnchorAppStatus) {
        // ...
    }

    override fun onAnchorAppUpgrade(context: Context, isUpgrade: Boolean) {
        // ...
    }
}
```

Compliance Check

There is no explicit compliance check interface in the SDK for Kotlin. An implicit compliance check is run whenever the application, or a service, starts and this applies even to background execution.

Inactivity Timer Reset

The following code snippet illustrates how to reset the user inactivity timer explicitly.

```
// ... In the Intent handling, for example.

// Reset the user inactivity timer, in case inactivity wipe is configured.
ApplicationLifecycleUtil.resetInteractionTime()

// ...
```

The SDK resets the user inactivity timer implicitly whenever the application comes to the device foreground. An explicit reset is required only when user interaction takes place without the application coming to the device foreground, such as in an Android notification.

Summary

To integrate this feature into a Workspace ONE Kotlin application:

- Ensure that `onClearAppDataCommandReceived` is implemented.
- If the user can interact without the application coming to the device foreground, call `resetInteractionTime` whenever they do so.

This concludes integration with the SDK for Kotlin.

Programming Interface for Swift

The following parts of the SDK programming interface for Swift are relevant to this feature.

Initialization

The following code snippets illustrate the general way to initialize the SDK, and handle inactivity wipe.

```
Class ControllerDelegate: AWControllerDelegate {
    func controllerDidFinishInitialCheck(error: NSError?) {
        // This method is invoked when the SDK has finished initialization.
        //
        // Run application code that is dependent on SDK start-up from here.
        //
        // In the case that the need for a data wipe is determined during start-up,
        // this method is invoked later than controllerDidWipeCurrentUserData, below.
    }

    func controllerDidWipeCurrentUserData() {
        // This method is invoked after the SDK has wiped data, for example
        // following a failed inactivity check.
        //
        // Run application code to do the following from here:
        //
        // -   Delete any data that was encrypted with a key that has been deleted.
        // -   Delete any data that wasn't encrypted.
        // -   Delete any data that was encrypted with a key from elsewhere than the SDK.
    }
}

// At start, for example in the didFinishLaunchingWithOptions implementation:
AWController.clientInstance().delegate = controllerDelegate;

// Next line will initialize the SDK. Initialization processing by the SDK
// includes an inactivity check.
AWController.clientInstance().start()
```

Note that there isn't a mode to initialize the SDK without making a compliance check.

Compliance Check

The following code snippets illustrate how to make an explicit compliance check.

```
ComplianceEvaluationController().evaluateComplianceStatus { (status) in
    if status == .nonCompliant {
        // The application instance isn't in compliance, for example, the inactivity
        // check failed. Encryption keys and credentials have been wiped.
        //
        // Run application code to do the following from here:
        //
        // -   Delete any data that was encrypted with a key that has been deleted.
        // -   Delete any data that wasn't encrypted.
        // -   Delete any data that was encrypted with a key from elsewhere than the SDK.
    }
}
```

Note that an implicit compliance check is run whenever the SDK is initialized. The explicit check interface supports applications that sometimes access data without

starting the SDK, for example in background.

Inactivity Timer Reset

There is no explicit inactivity timer reset interface in the SDK for Swift. The SDK resets the timer implicitly during initialization, after checking for inactivity. The application must initialize the SDK to open any of its user interface, for example as part of an App Extension, so an explicit reset wouldn't ever be needed.

Summary

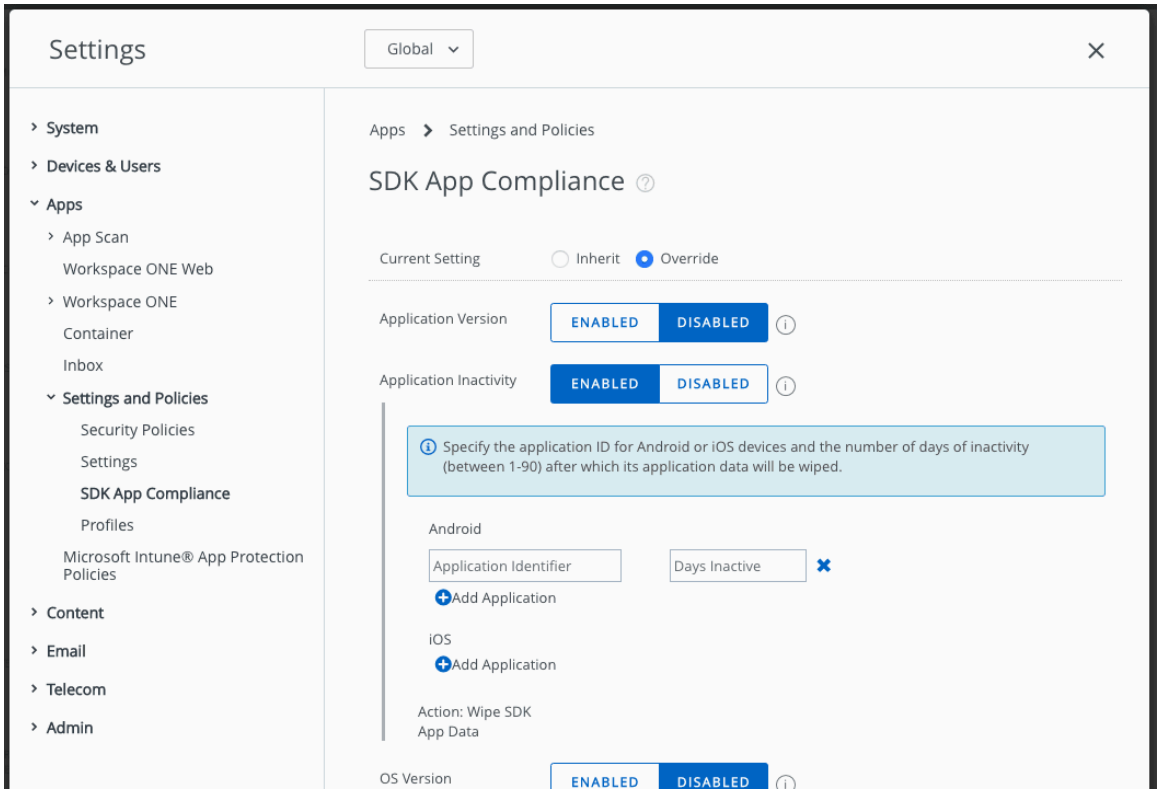
To integrate this feature into a Workspace ONE Swift application:

- Ensure that `controllerDidWipeCurrentUserData` is implemented.
- If the application accesses data without initializing the SDK, add a call to `evaluateComplianceStatus` wherever it does so.

This concludes integration with the SDK for Swift.

Appendix: Console User Interface

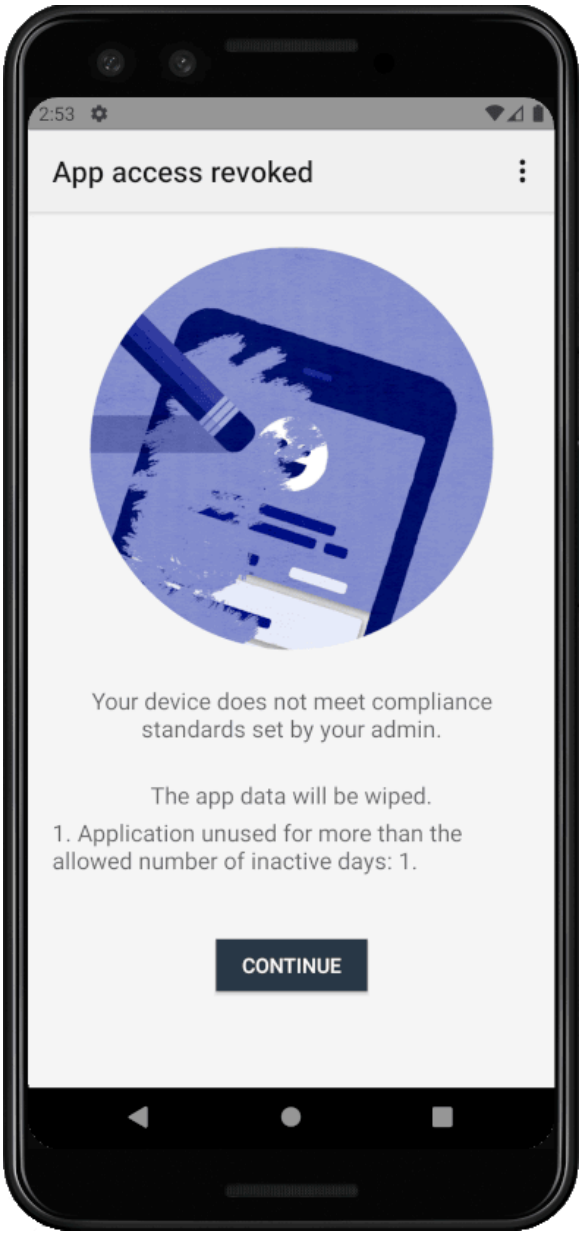
The following screen capture shows this feature's configuration in the management console.



Screen capture 1: Console User Interface

Appendix: User Interface for Android

The following screen capture shows the message displayed to the end user after an inactivity wipe.



Screen capture 2: Wiped Message for Android

Appendix: User Interface for iOS

The following screen capture shows the message displayed to the end user after an inactivity wipe.

Screen capture 3: Wiped Message for iOS

Document Information

Revision History

The following table shows the revision history of this document.

Date	Revision
23 Jul 2020	Initial Publication.
12 Feb 2025	Updated License.
04 Aug 2025	Document update.

License