Shared Device Support

The VMware Workspace ONE® platform has a Shared Device enrollment feature. The assigned end user of a device that is enrolled as shared can be changed without repeating the enrollment procedure.

The state and assignment of a Shared Device can be observed by mobile applications that have integrated the Workspace ONE mobile software development kit. This enables application developers to support specific Shared Device requirements and optimization.

Shared Device support relies on the following platform components.

- Workspace ONE Unified Endpoint Manager (UEM) console.
- Workspace ONE Intelligent Hub mobile application.
- Workspace ONE mobile Software Development Kit (SDK).

Table of Contents

Introduction

The Workspace ONE platform has a Shared Device enrollment feature. A device that has been enrolled as shared doesn't have a permanent association with a single end user. Instead, the device has a sequence of sessions in which it could be associated with different end users.

Shared Device sessions are started and finished by end users, who log in and log out at the device. There is only a single concurrent session; the current end user must log out before the next end user can log in.

The enrollment of a Shared Device persists between individual end user sessions. This means that, for example, mobile applications on the device aren't uninstalled when the current end user logs out.

Mobile applications that have integrated the Workspace ONE mobile SDK can observe and respond to sessions starting and finishing. This enables application developers to support specific Shared Device requirements and optimization.

The description here is intended for application developers and designers. For a full description of the Shared Device feature, see the product documentation, for example:

- docs.vmware.com/…shared_devices_overview.html

Background

To understand the types of requirement that can be met by application Shared Device support it is necessary to understand something about how Workspace ONE device set-up works.

Workspace ONE device set-up in general includes the following steps.

1. Establishment of a trusted communication channel between a device and a UEM instance.
2. Utilization of the communication channel to take a number of device management actions.
3. Establishment of a relationship between the device and a specific end user account in the UEM.

In the default case, all those steps happen in a single interaction, and none can be reversed individually. In the Shared Device case, set-up is divided between multiple interactions, the last of which can be reversed.

Description

Workspace ONE Shared Device set-up works as follows.

- End user roles are configured in the management console.

  There is one specific end user role that relates to this feature: the *Staging User*.

- Shared devices are enrolled.

  A new or reset device is enrolled by a Staging User. This is done in the usual way, by installing the Hub application on the device, and then entering a set of enrollment credentials.

  The credentials are validated, then the Hub installs a device management profile. Installation of the profile establishes a channel for the following types of action.

  - Any device restrictions associated with the profile are pushed to the device and enforced.
  - Electronic certificates and other credentials mandated by the profile are generated.
  - Any mobile applications associated with the profile are pushed to the device and installed automatically.

  When all these actions are finished, enrollment is complete.

  For a default end user, the completion of enrollment would mean that the device and applications were ready to use, but this isn't the case for a Staging User.

  The completion of enrollment by a Staging User, puts the Hub and managed applications into an unassigned state, referred to as the *checked-in* state, or sometimes the *staged* state.

  In the checked-in state:

  - The Hub shows only a prompt for credentials.
  - Managed application user interfaces are blocked.

- Enrolled devices can be checked out and checked in.

  End user credentials can be used to log in to a device that is in the unassigned checked-in state. The credentials would be entered in the Hub application on the device.

  When correct credentials have been entered, the Hub and managed applications enter an assigned state, referred to as the *checked out* state. This is the start of the user's session. In the checked-out state, the mobile applications on the device can be used as usual.

  When the user finishes their session, they log out in the Hub application.

For a screen capture showing the log-out option, see the Appendix: Mobile User Interface.

When the user logs out, the Hub and managed applications then return to the unassigned checked-in state. From there, another user could log in, and the device would again enter the checked-out state and start a new session.

See also the Appendix: Shared Device User Roles Diagram and the Appendix: Shared Device State Diagram for UML representations of some of the above.

Notes

The Shared Device feature has the following limitations.

- Use of the Hub application is mandatory.
- Use of managed profiles is mandatory, i.e. Mobile Device Management (MDM).
- Check-out requires a connection to the management console.

  Validation of credentials for check-out by a new user requires a connection to the UEM. A device must be online to enter the checked-out state.

Also note the following details.

- An application-level passcode could be required.

  When an end user checks out a device, they might be required to set an application-level passcode. This depends on the security policies configured in the UEM. This would be distinct to any device passcode, which could also be required by security policies.

  There will be only a single passcode for all applications if Workspace ONE Single Sign-On (SSO) for apps is in use.

- Offline unlock switches to Hub.

  The user interface of a managed application will be locked by the SDK if the user is inactive for an extended period. Inactivity lock-out is a general Workspace ONE SDK feature and isn’t specific to shared devices.

  If the user attempts to unlock a managed application on a Shared Device when the device is offline, the user interface will first switch to Hub. If the device is checked out, then the user interface will switch back to the application. If the device is checked in, then the user interface won’t switch back from Hub.

- Check-in requires unlock.

  A device cannot be checked in when the user interface is locked, for example because inactivity was detected. In that case, the end user must authenticate in order to check in the device.

- Check-in isn’t broadcast.

  As described above, to end their session, a user opens the Hub application and selects to log out. When that happens, the Hub immediately enters the checked-in state. However, this transition cannot be broadcast to other applications. Other applications will enter the checked-in state only when launched.

- Check-out isn’t broadcast.

  As described above, to start a session, the user opens the Hub application and logs in. When that happens, the Hub immediately enters the checked-out state. However, this transition cannot be broadcast to other applications. Any applications that had been launched when the device was in the checked-in state will enter the checked-out state only when terminated and relaunched.

- The Staging User can be an end user.

  A Staging User can use the applications on the device as an end user, if they authenticate a second time and check out the device.

Application Support

An application that has integrated the Workspace ONE mobile SDK can support specific Shared Device requirements and optimization.

Programming Interface Overview

The SDK programming interface enables application support for shared devices as follows.

- Current End User Identity.

  The SDK makes available the identifying details of the current assigned end user. These details come from the UEM. This interface isn't specific to shared devices.

  This interface is only available after the SDK has been initialized.

- Shared Device Detection.

  There is a flag in the SDK for whether the device is enrolled as shared.

- Check-Out Notification.

  The SDK invokes a specific user-changed callback to notify the application code that the device has been checked out. The invocation could be at different times, as follows.

  - If there is a check-out while the application is running in background, the callback will be invoked when the application returns to the foreground.
  - When the application starts cold and initializes the SDK, the callback will be invoked if there has been a check-out since the application was last running.

  It is possible that the device is checked out, and checked in, multiple times without an opportunity for the user-changed notification to be invoked. In that case, when there is an opportunity, there will be only one invocation.

- Check-In Notification.

  The SDK invokes the general callback for wiping user data to notify the application code that the device has been checked in. Invocation takes place opportunistically, same as for the Check-Out Notification, above.

  When a check-in is detected, the SDK deletes all its application data encryption keys. In the SDK for Swift, this includes the keys that underlie the `AWController` `encrypt` programming interface, for example. Any data encrypted by that interface cannot be decrypted after a check-in.

  The callback will be invoked after the keys have been deleted.

- Checked-in Start Blocked.

  Initialization of the SDK will fail if the device isn't checked out to any end user. The SDK notifies the application code when this occurs, by passing an error to the general initialization-finished callback.

For details and code snippets illustrating the programming interfaces, see Programming Interface for Swift.

Use Cases

The following use cases can be read as a starting point for requirements analysis of Shared Device support.

Use Case: No Shared Device Support

By default, an application will have no Shared Device support.

Note that the SDK posts the same type of notification for the following events.

- Enterprise wipe command was received from the UEM, for example if the user reported the device lost or stolen.
- Current end user checked in the device.

On check-in, therefore, an application that doesn't support Shared Device will delete all local stored data.

There are still some user experience improvements in this case. For example, a device can be re-assigned without the need to re-install mobile applications.

Use Case: Retain Local Data

A generic use case for Shared Device support is the retention of local application data on check-in.

In order to retain local data, the application must ignore data wipe notifications, to some extent. Specifically:

- The application will still delete any local data that it protected with an encryption key from the SDK.
- The application won't delete some other data.

Data that isn't deleted is, in principle, vulnerable to an attacker with access to the device. Retention could be limited to data that isn't sensitive, or the retained data could be protected by other mechanisms. For example:

- Data could be encrypted with a key that isn't stored on the device but to which an authenticated user would have access.
- Offline access could be deactivated in the UEM policy configuration.
- Device protection could be used, instead of container protection.

The procedure for lost and stolen devices could be to do one of the following.

- Wipe or lock the whole device from the UEM.
- Uninstall the application remotely, also from the UEM.

The following sections discuss more specific use cases based on retention of local data.

Use Case: Retain Preferences

In some use cases:

- The application supports preferences or other settings for personalization by the end user.
- There is a relatively small number of end users, so that the same end user quite often checks out the same Shared Device.

If no Shared Device support is implemented, all personalization would be reset on check-in. The end user would then have to re-apply their preferences every time they check out a device.

To support these use cases, the application developer could implement the following.

- Store the set of preference selections made by the current end user on the device, unencrypted.
- Label the set with the current end user identity.
- On check-out, if there are already a set of stored preferences for the new end user, apply them automatically.

When a user checks out a device, if they have checked out that device before, their preferences are already selected.

Use Case: Cache Common Data

In some use cases:

- There is a substantial amount of data that is common to all end users.
- The data isn’t completely static and must be synchronised to the device.

This use case could apply to, for example, procedure documents, design blueprints, and inventory. If Shared Device support isn’t implemented, all common data would be deleted on check-in, and then have to be downloaded again on check-out.

To support these use cases, the application developer could implement the following.

- On first check-out, download common data to the device. Store the data unencrypted.
- On subsequent check-out, or opportunistically, download and synchronise updates to the common data. This should be quicker than downloading all the

common data.

- On check-in, delete any data encrypted by the SDK but retain the stored common data.

When a user checks out the device, the application would already have the common data.

Use Case: External Encryption

In some use cases, each end user has an encryption key that is stored off the device.

To support these use cases, the application developer could implement the following.

- On check-out, obtain the encryption key. Depending on the external storage type, this could involve the user presenting a second factor, or logging in to an external server for example.
- Retain the key in memory only.
- Use the key to encrypt local data during the session.
- On check-in, discard the key, but retain any data encrypted by the key.

The retained data on the device remains protected after check-in, and couldn't be exposed even to another end user.

Use Case: Shared Device Mode

Some Shared Device support implementations introduce security vulnerabilities whose remedies aren't necessarily applicable outside the Shared Device use cases. For example, the retention of unencrypted data after a wipe notification is a vulnerability that is remedied by adoption of MDM device wipe as procedure. That remedy can only be applied to managed devices.

Some Shared Device support implementations introduce undesirable user experience elements, such as the need for a second authentication factor.

These concerns can be addressed by implementing a shared device mode, as follows.

- On start-up, check if the application is running on a Shared Device.
- If it is, then run Shared Device supporting implementations.
- Otherwise, run default implementations.

For example, the default implementation could be to protect all local data with an application key managed by the SDK, and delete all application data when a device wipe is received.

In this way, a single application executable supports both Shared Device and default deployments.

Integration

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

Compatibility

Before you begin integration, ensure you have access to compatible versions of software. The following table shows the version numbers of the Workspace ONE components in which this feature first became available.

Software	Available
Workspace ONE SDK for iOS	21.1
Workspace ONE management console	9.7*

Version 9.7* is the earliest supported UEM at time of writing. All current versions of UEM have the Shared Device option.

Account Configuration

First, configure the necessary end user accounts in the Workspace ONE management console.

Set up three end user accounts:

Account	Device usage
Staging user	Enroll.
First shared device user	Check out. Use applications. Check in.
Second shared device user	Check out. Use applications. Check in.

Some instructions for setting up these accounts are given below. These are intended for application developers or others wishing to try out Shared Device support. Full documentation can be found in the UEM product documentation.

User Security Type

The instructions specify configuration of UEM *Basic* user accounts.

Basic user accounts exist only in the UEM and aren’t linked to, for example, users in a Lightweight Directory Access Protocol (LDAP) directory. For that reason they could be the easiest type of account to set up for development purposes.

Every UEM supports Basic users by default. In some deployments, Basic users will be the only option.

Basic users authenticate by entering a user name and password at the device.

Set up a Staging User for shared device enrollment

To set up a Staging User that can enroll a shared device, proceed as follows.

1. Open the Workspace ONE management console in a web browser and log in.

   This opens the dashboard.

2. Select an organization group.

   By default, the Global group is selected.

3. From the dashboard, navigate to: Accounts, Users, List View.

   This opens a list of user accounts.

4. Select: Add User.

   This opens the Add/Edit User dialog.

5. Select Security Type: Basic.

6. Enter the required general details.

   For development purposes, you might want to use values like the following.

   - Username: “stage” plus your enterprise user name.
   - Password and confirm: Same value as Username.
   - First Name: Your first name.
   - Last Name: “Staging”.
   - Email Address: Your enterprise email address.

   (Including your own name here can help other users of your developer environment get in touch with the correct person if they wish to make changes.)

   When you have finished, select the Advanced tab.

7. On the Advanced tab, expand the Staging section.

8. Set the option Enable Device Staging: Enabled.

9. Set the option Multi User Devices: Enabled.

10. Click Save to create the user.

    The list view is displayed again, now with the Staging User that you just created.

For a screen capture showing the configuration of a staging user, see the Appendix: Management Console User Interface.

If you made a mistake in the instructions, or want to change anything, you can edit the user from the list view by clicking the pencil icon next to their name.

Set up a first shared device user

To set up a user that can check out an enrolled device, proceed as follows.

1. Open the Workspace ONE management console in a web browser and log in.

   This opens the dashboard.

2. Select an organization group.

   By default, the Global group is selected.

3. From the dashboard, navigate to: Accounts, Users, List View.

   This opens a list of user accounts.

4. Select: Add User.

   This opens the Add/Edit User dialog.

5. Select Security Type: Basic.
6. Enter the required general details.

   For development purposes, you might want to use values like the following.

   - Username: “cou1” plus your enterprise user name.
   - Password and confirm: Same value as Username.
   - First Name: Your first name.
   - Last Name: “CheckOut 1”.
   - Email Address: Your enterprise email address.

   (Including your own name here can help other users of your developer environment get in touch with the correct person if they wish to make changes.)

7. Click Save to create the user.

The list view is displayed again, now with the user that you just created. If you made a mistake in the instructions, or want to change anything, you can edit the user from this screen by clicking the pencil icon next to their name.

Set up a second shared device user

Follow the instructions under Set up a first shared device user, but using the number 2 instead of the number 1 in the general details.

Device Transitions

After setting up the users, see above, your developer device can be taken through Shared Device transitions.

Device Preparation

Prepare as follows.

- You will need a physical device to try out this feature; a simulated or emulated device cannot be used.
- Don’t use a device that is already enrolled with a Workspace ONE console. A device can be unenrolled by removing or resetting the Hub application on the device, and uninstalling any associated applications.

Device Enrollment

Enroll the device as follows.

1. Install the Workspace ONE Intelligent Hub application.
2. Enroll with the staging user credentials.

Enrollment processing might take a few minutes. Certificates could be installed, and applications, and device management restrictions could be applied to the device.

After enrollment has finished, a prompt for credentials will again be shown by Hub. For example, a prompt for the UEM Organizational Group identifier to be entered could be shown. This will be the first step of device check-out.

Device Check Out and Check In

The following steps will demonstrate the Shared Device feature.

1. Open the Hub application and enter the credentials of the first shared device user created above. The device is now checked out to the first user.
2. Open the Hub application and select to log out. The device is now checked in.

   For a screen capture showing the log-out option, see the Appendix: Mobile User Interface.
3. Open the Hub application and enter the credentials of the second shared device user created above. The device is now checked out to the second user.

Repeat this sequence, completely or partially, to test your application’s handling of Shared Device transitions.

Programming Interface for Swift

The following parts of the SDK programming interface for Swift are relevant to this feature. See Application Support for an overview of the programming interface.

Shared Device Detection

The following code snippet illustrates how to check whether the device is enrolled as shared.

```
DeviceInformationController.sharedController.fetchApplicationAssignmentStatus {(
    applicationStatusInformation: ApplicationStatusInformation?,
    sharedDeviceInformation: SharedDeviceInformation?,
    error: Error?
) in

    if (
        sharedDeviceInformation != nil && information.isDeviceStagingEnabled
    ) {
        // Code to track that the device is shared goes here.
    }
    else {
        // Code to track that the device isn't shared goes here.
    }

})
```

Current End User Identity

The following code snippet illustrates how to access the current end user identity details. The programming interface is asynchronous and based on a callback.

```
UserInformationController.sharedInstance.retrieveUserInfo {(userInformation, error) in
    guard
        let userInformation = userInformation,
        error == nil
    else {
        // Code to handle the error goes here. For example, log and display to the user
        // error.debugDescription
        return
    }

    // Code to store or check the details goes here. The following properties of
    // the userInformation object can be used.

    // Username
    userInformation.userName

    // Group ID
    userInformation.groupID

    // Email Address
    userInformation.email

    // Full Name
    userInformation.firstName
    userInformation.lastName

    // Domain
    userInformation.domain

    // User ID
    userInformation.userIdentifier
}
```

Initialization and Notifications

The following code snippets illustrate the general way to initialize the SDK and receive notifications related to Shared Device support.

```
Class ControllerDelegate: AWControllerDelegate {
    func controllerDidWipeCurrentUserData() {
        // This method is invoked after the SDK has wiped data, for example
        // after detecting that the device has been checked in.
        //
        // Code to do the following goes here:
        //
        // -   Delete any data that was encrypted with a key that has been deleted.
        //
        // There is no need to delete data that is protected in some other way.
    }

    func controllerDidDetectUserChange() {
        // This method is invoked opportunistically when the SDK detects that the
        // device has been checked out.
        //
        // Code to do the following goes here:
        //
        // -   Set a flag that the user has changed.
        //
        // This callback will be invoked before the did-finish callback, below. At
        // that stage in the SDK startup, the current end user identity, see above,
        // won't be available.
    }

    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.
        //
        // For example:
        if (
            error != nil &&
            error.domain == AWSDKErrorDomains.setup &&
            error.code == AWSDKSetupError.sharedDeviceNotCheckedOut
        ) {
            // Code to handle the specific case that the application has been started
            // on the device but no user has logged in goes here, if needed.
        }

        // Also, code to do the following could go here:
        //
        // -   Check if the user-changed flag has been set, see under
        //     controllerDidDetectUserChange, above.
        // -   Check for a local cache of data for the new end user, if any.
        // -   Retrieve data specific to the new end user, if any.
        // -   Apply stored or retrieved preferences for the new end user, if any.
    }
}

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

// Next line will initialize the SDK.
AWController.clientInstance().start()
```

The expected order of notification callback invocation during initialization is:

1. `controllerDidWipeCurrentUserData` if applicable.
2. `controllerDidDetectUserChange` if applicable.
3. `controllerDidFinishInitialCheck`.

Appendix: Shared Device User Roles Diagram

The following diagram represents the user roles that are relevant to the Shared Device feature as a UML use case diagram.

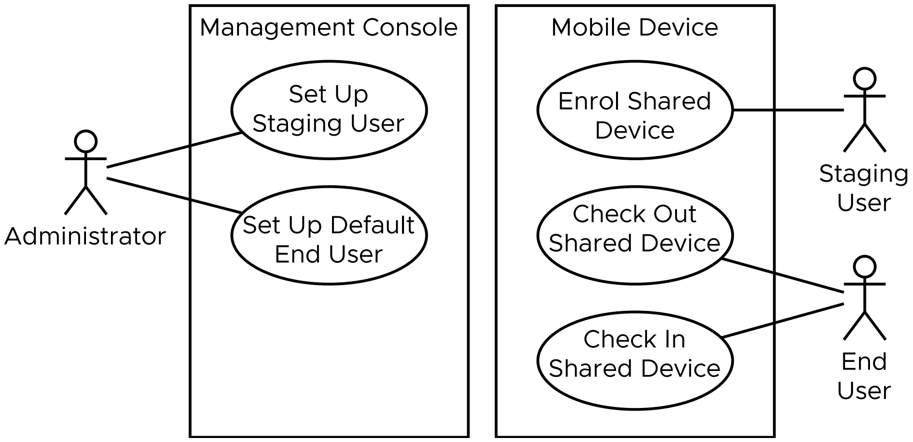


Diagram 1: Shared Device User Roles

Appendix: Shared Device State Diagram

The following diagram represents the transitions that are relevant to the Shared Device feature as a UML state machine diagram.

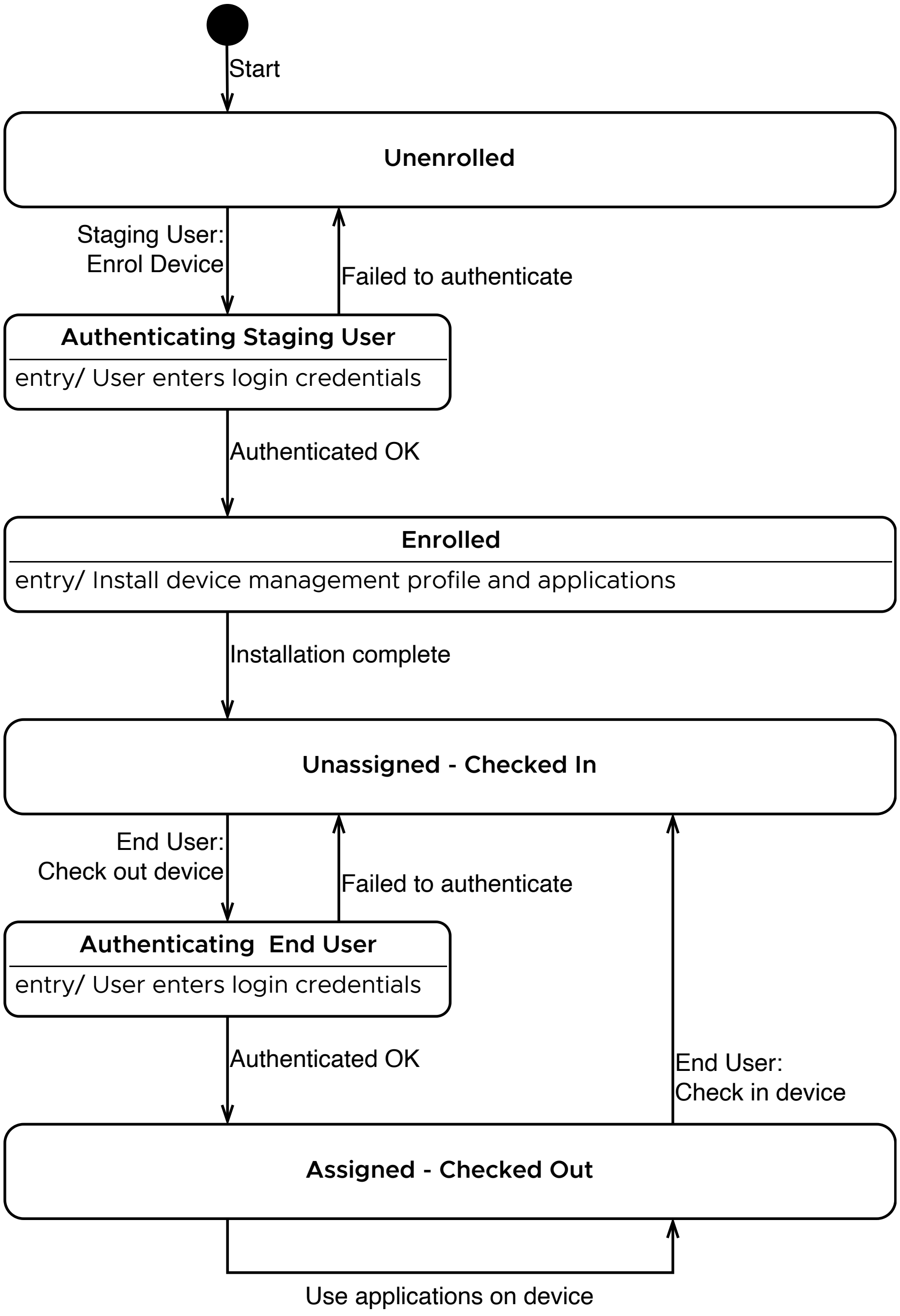
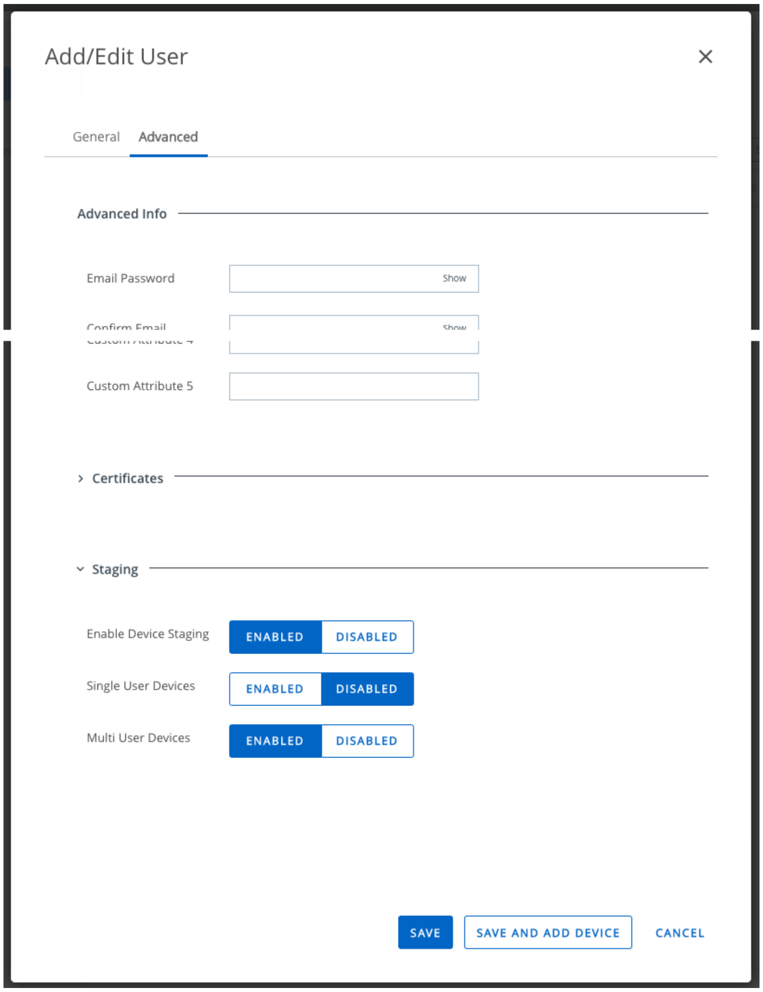


Diagram 2: Shared Device State

Appendix: Management Console User Interface

The following screen capture shows the configuration of a Staging User in the Workspace ONE management console. (Part of the screen isn't shown, so that the capture image fits on one page.)



Add/Edit User

General | Advanced

Advanced Info

Email Password

Confirm Email

Custom Attribute 4

Custom Attribute 5

› Certificates

⌄ Staging

Enable Device Staging	[✓ ENABLED] DISABLED
Single User Devices	ENABLED [✓ DISABLED]
Multi User Devices	[✓ ENABLED] DISABLED

Screen capture 1: Console User Interface Staging User Set Up

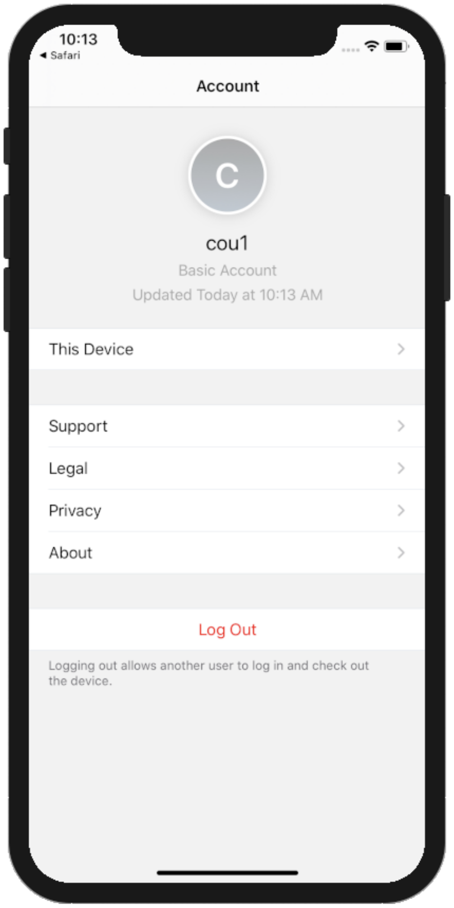
The option for Multi User Devices is selected. A device enrolled by this account will become a Shared Device that can be checked out and checked in.

Selecting the option for Single User Devices would make the staging user into a kind of assistant account. A device enrolled by the account would subsequently become permanently associated to the first user that authenticated after enrollment.

Appendix: Mobile User Interface

The following screen capture shows the Shared Device check-in option in the Workspace ONE Intelligent Hub application.

In this screen capture, the user name is a dummy value, “cou1”, and not a real user name.



cou1
Basic Account
Updated Today at 10:13 AM

Logging out allows another user to log in and check out the device.

Screen capture 2: Hub for iOS with option to log out

Document Information

Revision History

21jan2021	Initial Publication.
20aug2021	Updated links.

Legal

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877–486–9273 Fax 650–427–5001 www.vmware.com