# ketterhead peelbackIntroduction

The Shared Device Module is designed to switch the assigned user of a device within Workspace ONE UEM (WS1). Assignment of a device to a user is important for Service Desk / Inventory functions and also allows the Workspace ONE Hub (Catalogue) app to work.

The module also moves the device to a designated Organization Group (OG) or will keep the device in the current OG based upon the desired configuration.

The script uses API calls to the WS1 API server(s) to determine if the device is set as ‘Corporate – Shared’. If it is not, the script exists, and no action occurs. This means the module can be deployed widely.

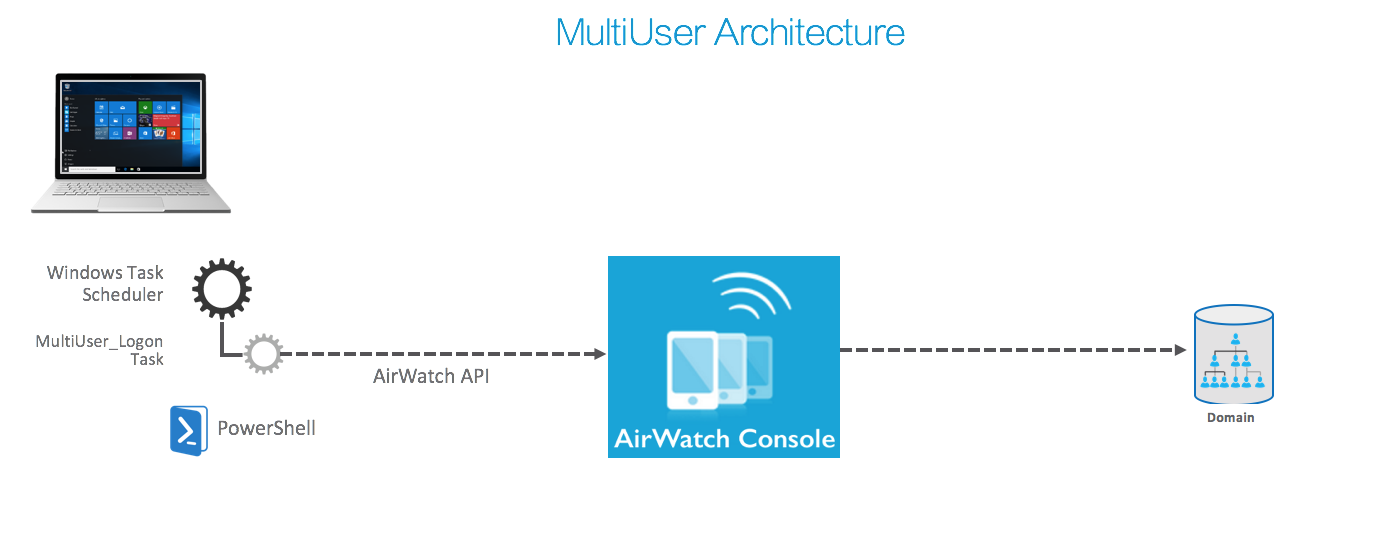
The module has also been designed with the following scenarios in mind:

* Standard User logon
* Screen locked, second user logon
* Screen unlock, user logon
* Device moved to designated OG (configurable)
* Device kept in current OG (configurable)

The module is provided as-is, with best effort support and is based upon components included in the Custom Device Inventory found on <https://code.vmware.com>.

# How It Works

The Shared Device Module for Windows 10 is relatively simple – outlined below are the steps and components used:



1. A Scheduled Task runs ‘At Logon’ as well as at ‘Workstation Unlock from any user’ and runs the AWLogon.ps1 script.
2. The script reads the shared.config file in the Shared folder.
3. The AWLogon.ps1 script then calls functions to determine the following:
   1. If the device is set as ‘Corporate – Shared’. If yes:
      1. Who the current logged on user (Active and Console) is
      2. Does the current logged on user match the assigned user:
         1. if no, assign the current logged on user to the device
      3. If the shared.config LogonGroup is set to “Current”, then leave the device in current OG. If not:
         1. Determine current OG
         2. Is the device in the LogonGroup OG, if yes, leave device in current OG. If not,
            1. move the device to configured LogonGroup OG
4. If No, do nothing

# Implementing the Module

There are multiple steps and multiple files to be updated for specific environment configuration. The extracted folder should contain the following files:

Graphical user interface, table

Description automatically generated with medium confidence

## Step 1: Configure the service account for the API connection

A WS1 Administrator account is used by devices for secure communication to WS1 API Servers. There are only a few admin rights needed with the primary ones being read, and write permissions to the device.

For testing purposes, a WS1 Administrator account with Console Administrator Role assigned can be utilised. Moving to production, I recommend selecting an existing WS1 Role or creating a custom Role with Read & Write permissions to Devices.

Next, create a new Basic or Directory WS1 Administrator account with the appropriate WS1 Role at the top-level organization group. Record the username and password and then Base64 encode them using the format:

Username:Password

<https://coderstoolbox.net/string/#!encoding=base64&action=encode&charset=utf_8> is useful for this task.

## Step 2: Update the api.config file to match your environment

The api.config file found in the Shared folder of the module contains information to connect securely to the REST API interface of the WS1 environment. This file is read during installation, encrypted and stored in the Shared folder to prevent credentials being compromised. The unencrypted api.config file along with the source ZIP file deployed with WS1 Software Delivery, are also deleted during installation.

The following process should be followed to edit the api.config file and update the necessary attributes. The api.config file is shown for completeness below:

{

"ApiConfig": {

"DeviceId": "",

"OrganizationGroupId": ORGID,

"Server": "https://server.awmdm.com",

"ApiKey": "APIKEY",

"ApiAuth": "Basic BASE64\_USERNAME:PASSWORD",

"OrganizationGroupName": ""

}

}

* Update the **OrganizationGroupId** attribute with the number of the highest level OG. You can find this # on the Organization Groups detail page URL.  
  A screenshot of a cell phone

  Description automatically generated
* Update the **Server** attribute to your WS1 API Server FQDN, eg:  
    
  <https://APISERVER.awmdm.com>
* Update the **APIKey** attribute with your API Key - All Settings 🡪 General 🡪 Advanced 🡪 API 🡪 REST. You can utilise the default AirWatchAPI key or create another.
* Update the **APIAuth** attribute with the Base64 encoded username:password from Step 1. Ensure the Word “Basic “ preceeds the Base64 encoded username:password.

## Step 3: Configure your OGs for shared devices

Configure the **shared.config** file found in the UserManagement folder of the module. This configuration file contains information used to checkout/checkin the device once a user assignment has been made. It is advised to utilise OGs below the Production OG if Tier 1 or common applications are assigned to the OG layer with the intent of deploying to all devices. The action of moving OGs will force applications not assigned to that OG or inherited from a higher OG to be removed.

The following process should be followed to edit in the shared.config file and update the necessary attributes. The shared.config file is shown for completeness below:

{  
 "SharedConfig": {  
 "LogoffGroup":"sharedunassigned",  
 "LogonGroup":"current",  
 "StagingUser":"staging@sharedunassigned.com"  
 }  
}

* **Create** the **LogoffGroup** and **LogonGroup** OGs within the WS1 Console
* Update the **LogoffGroup** and **LogonGroup** attributes with the Group ID of the WS1 Organization Group  
  A screenshot of a cell phone

  Description automatically generated
* Update the **StagingUser** attribute with the email address of a basic user  
  A screenshot of a cell phone

  Description automatically generated

The best practice for this solution is to have two nested OGs for the devices, so that the script can initiate an update of the smart groups the device is a member of. It is also possible to have non-nested OGs. **At present, the functionality to move a device into the LogoffGroup has been disabled.**

Note, that whilst the LogoffGroup is not utilised at present, the OG must exist.

Setting the **LogonGroup** to **“Current”** will keep the device in the same OG, effectively only changing user assignment. This configuration will reduce the number of API calls, simplify deployment in complex OG hierarchies, and reduce the risk of applications being uninstalled and installed due to differing assignment at the OG layer.

## Step 4: Update the setup.manifest file to match your environment

The setup.manifest controls the install and uninstall of the package.

* Update the **Modules.Version** to match the version of this module. This attribute is created in the HKLM:\SOFTWARE\AIRWATCH\ProductProvisioning registry key and will be used as the “**When to Call Install Complete**” Registry test described in Step 5.
* Update the **Modules.DeleteFiles** .\\SharedDeviceModulev2801.zip to match the name of the ZIP file you will upload to Workspace ONE.  
  These files are deleted during install in order to reduce the risk of the APIKey and APIAuth entries being used.

{"Modules":[

{"Name":"Shared",

"InstallLocation":"C:\\ProgramData\\Airwatch\\Shared",

"Version":"2801",

"Manifest":[

{"CopyFiles":

{"Destination":"$InstallLocation",

"From":".\\Shared\\\*"

}

},

{"CreateAccessFile":{

"Location":"$InstallLocation",

"SecurityLevel":0,

"AccessRules": [{

"AccessLogic": [

{"Group": "Users",

"Rule": "IN"},

{"User": "Administrator",

"Rule": "NOTIN"}

]

}]

}

},

{"DeleteFiles":[

".\\Shared\\api.config",

".\\SharedDeviceModulev2801.zip"

]},

{"CreateRegKeys":[

{ "Keys":[

{"LogPath":"C:\\ProgramData\\Airwatch\\Logs"},

{"SharedPath":"C:\\ProgramData\\Airwatch\\Shared"},

{"TestPath":"C:\\ProgramData\\Airwatch\\Shared"}

]

}

]}

]

},

{"Name":"AWLogon",

"InstallLocation":"C:\\ProgramData\\Airwatch\\UserManagement",

"Version":"1.0.0",

"Manifest":[

{"CopyFiles":

{"Destination":"$InstallLocation",

"From":".\\UserManagement\\\*"}

},

{"CreateTask":{

"Name":"MultiUser\_Logon",

"PSFile":"$InstallLocation\\AWLogon.ps1",

"TriggerType":"onUnlock"

}}

]

}

]

}

## Step 5: Deploy to devices

Package and deploy the module to devices.

* ZIP the contents of the module with the updated configuration files being sure not to zip the folder:  
    
  A screenshot of a social media post

  Description automatically generated
* Name the ZIP file **SharedDeviceModulev2801.zip as per Step 4**. This file is deleted during installation in order to remove the API and credential details from the device. If using a different name, update the **setup.manifest** file included in the ZIP and referenced during installation.
* Create a new Internal Native Application in WS1 Console – Apps & Books > Native > Add Application:
  + Upload new ZIP file
  + Set Name:  
      
    **Shared Device Module v2801**
  + On Files tab, set uninstall command line to:  
      
    **powershell.exe -ExecutionPolicy Bypass -File .\Uninstall.ps1**
  + On Deployment Options tab, Set install command line to:  
      
    **powershell.exe -ExecutionPolicy Bypass -File .\SetupEx.ps1**
  + Set Install Context:  
      
    **Device**
  + Set Admin Privileges:  
      
    **Yes**
  + Add criteria to determine when installation is complete:  
      
    **Registry Exists – HKLM\SOFTWARE\AIRWATCH\ProductProvisioning\SharedIVersion = [string]2801**
  + Assign to a SmartGroup or OG

## Step 6: Configure Devices as Corporate - Shared

Finally, configure devices as “Corporate – Shared”. The solution will only run on Corporate – Shared devices. To do this manually:

* From the WS1 console – Devices > List View > Click on the Device
* **More Actions** > **Edit Device**

Graphical user interface, application

Description automatically generated

* Change Device Ownership to “**Corporate – Shared**”

Graphical user interface, application

Description automatically generated

## Troubleshooting

Logs are created in the C:\ProgramData\Airwatch\Logs folder. By default, there is minimal logging.

If more logging is required, edit the **C:\ProgramData\Airwatch\UserManagement\AWLogon.ps1** script and set **$Debug = $True**