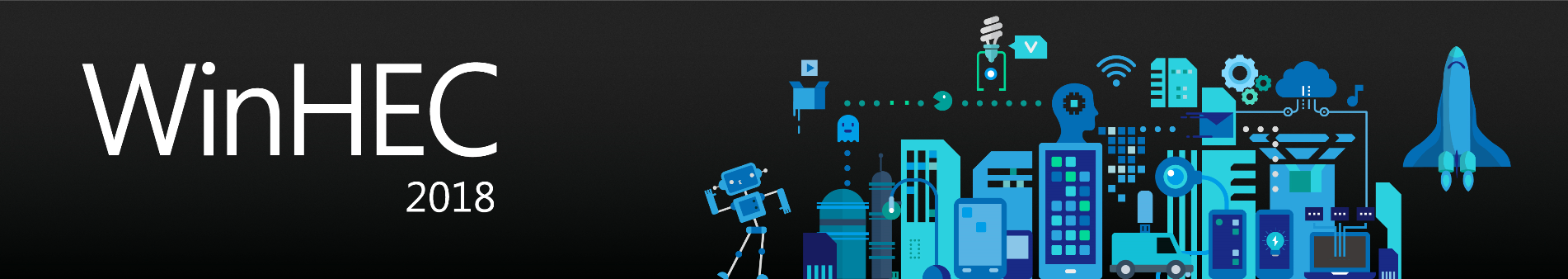
WinHEC

 Hands-on Lab

Optimizing Windows Devices for

Screen-on Battery Life

***Abstract*:** In this lab we dive into the Performance Power Slider UX which provides end users with more control of making trade-offs between performance and power and the Energy Estimation Engine in Windows 10 which provides energy usage data broken down by apps, services, and hardware components on retail devices that are in-use by end users.

Copyright (c) 2018 Microsoft. All rights reserved.

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice.

Contents

[Lab Goals 4](#_Toc527031427)

[Terminology 4](#_Toc527031428)

[Prerequisites 5](#_Toc527031429)

[Exercise 1 – Introduction to Battery User Experience in Windows 6](#_Toc527031430)

[Exercise 2A – Perf Power Slider: Opt out of Power Throttling 10](#_Toc527031431)

[Exercise 2B –Perf Power Slider: Viewing PPM settings in WPA 13](#_Toc527031432)

[Exercise 2C –Perf Power Slider: Viewing Overlay Settings using PowerCfg 15](#_Toc527031433)

[Exercise 2D – Perf Power Slider: Adding custom OEM settings 17](#_Toc527031434)

[Exercise 3A – Accessing Energy Estimation Engine (E3) data using PowerCfg 20](#_Toc527031435)

[Exercise 3B – Visualizing Energy Estimation Engine data in PowerView 21](#_Toc527031436)

[Exercise 3C – Collecting & Visualizing Energy Estimation ETW events in WPA 24](#_Toc527031437)

[Exercise 4 – Battery Life Assessments 26](#_Toc527031438)

[Exercise 5A – Verify Audio is Offloaded During FSVP 33](#_Toc527031439)

[Exercise 5B – Verify Low Refresh Rate Playback is Enabled During FSVP 37](#_Toc527031440)

[Exercise 5C – Verify MPO and iFlip are Enabled During FSVP 39](#_Toc527031441)

[Exercise 6A – Modify Power Settings with PowerCfg 41](#_Toc527031442)

[Exercise 6B –PowerCfg System Power Report For Optimizing Screen On Power 43](#_Toc527031443)

[Exercise 6C – PowerCfg Battery Report to Get Battery History & Stats 45](#_Toc527031444)

[Exercise 7 – Identify Apps Reducing the System Timer Resolution in WPA 47](#_Toc527031445)

[Annex I - Technical background 49](#_Toc527031446)

[Annex II – E3 Logs field descriptions 52](#_Toc527031447)

# Lab Goals

By the end of this hands-on lab, you will have a better understanding of the following battery life topics:

**Battery Life Features in Windows**

* Performance Power Slider UI
* Battery Saver UI
* Battery usage by app UI
* Energy Estimation Engine (E3)
* Low power audio/video playback

**Battery Life Assessments**

* Energy Efficiency assessments in the Windows Assessment Toolkit
* Windows Assessment Console (WAC)

**Optimization & Analysis Tools**

* PowerCfg and PowerView
* Windows Performance Recorder (WPR)
* Media eXperience Analyzer (MXA)
* Windows Performance Analyzer (WPA) in the Windows Performance Toolkit (WPT)

# Terminology

|  |  |
| --- | --- |
| Acronym | Definition |
| ETW | Event Tracing for Windows  Low overhead performance and quality tracing in retail builds |
| ADK | Windows Assessment & Deployment Kit |
| WAT | Windows Assessment Toolkit – performance, quality, & battery life assessments |
| WAC | Windows Assessment Console – used for viewing assessment results |
| WPT | Windows Performance Toolkit |
| WPA | Windows Performance Analyzer – visualize & analyze ETW logs |
| WPR | Windows Performance Recorder - capture ETW logs |
| MXA | Media eXperience Analyzer - Visualize & analyze ETW logs for media scenarios |
| FSVP | Full Screen Video Playback |
| E3 | Energy Estimation Engine |
| SRUM | System Resource Utilization Monitor |
| PPM | Processor Power Management |

# Prerequisites

To successfully complete the steps outlined in this lab, please review and prepare the following pre-requisites:

* Hardware: A laptop with a battery
* Assessment and Deployment Kit (ADK), including Windows Performance Toolkit and Media eXperience Analyzer
* Windows 10 ADK (1809): <https://developer.microsoft.com/en-us/windows/hardware/windows-assessment-deployment-kit>
* Microsoft Office (Word, Power Point, OneNote, Outlook, Excel)
* Netflix App in Microsoft Store

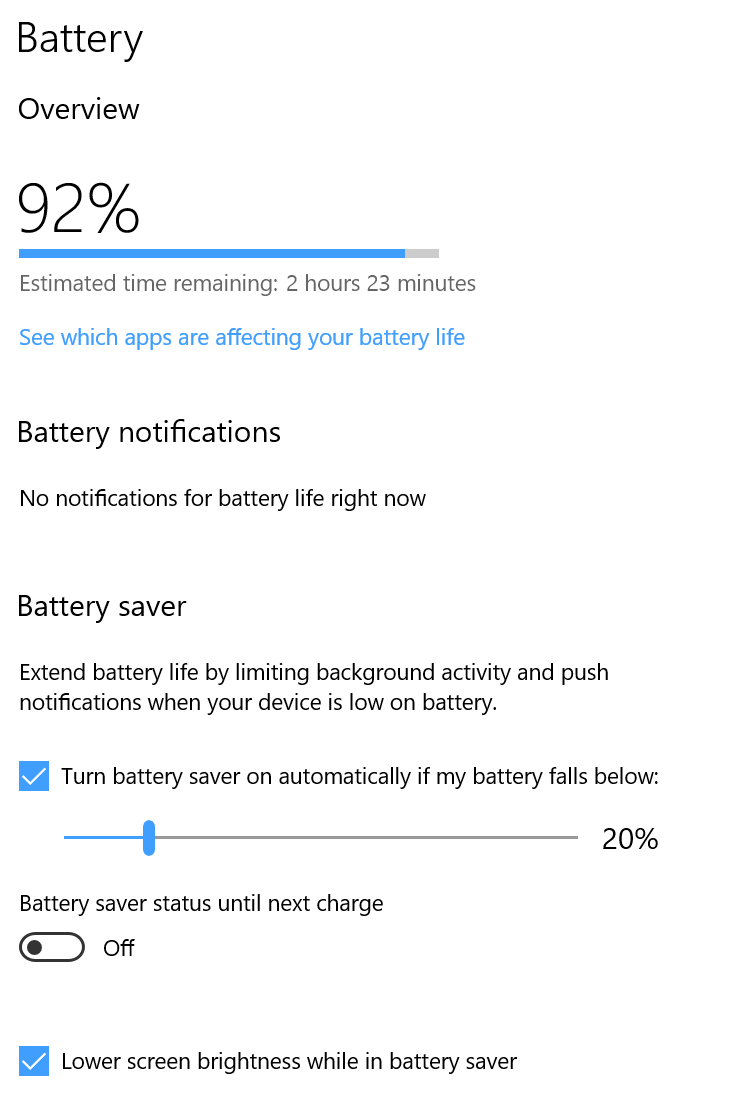
Additional technical reference and background can be found in the Annex I at the end of this document

# Exercise 1 – Introduction to Battery User Experience in Windows

**Estimated Time**: 5 minutes  
**Level**: 100  
  
In this exercise, you will learn more about the controls and types of data that Battery Saver & “Battery Usage by app” UI provides to end-users.

## Step 1 – Launch Battery Saver

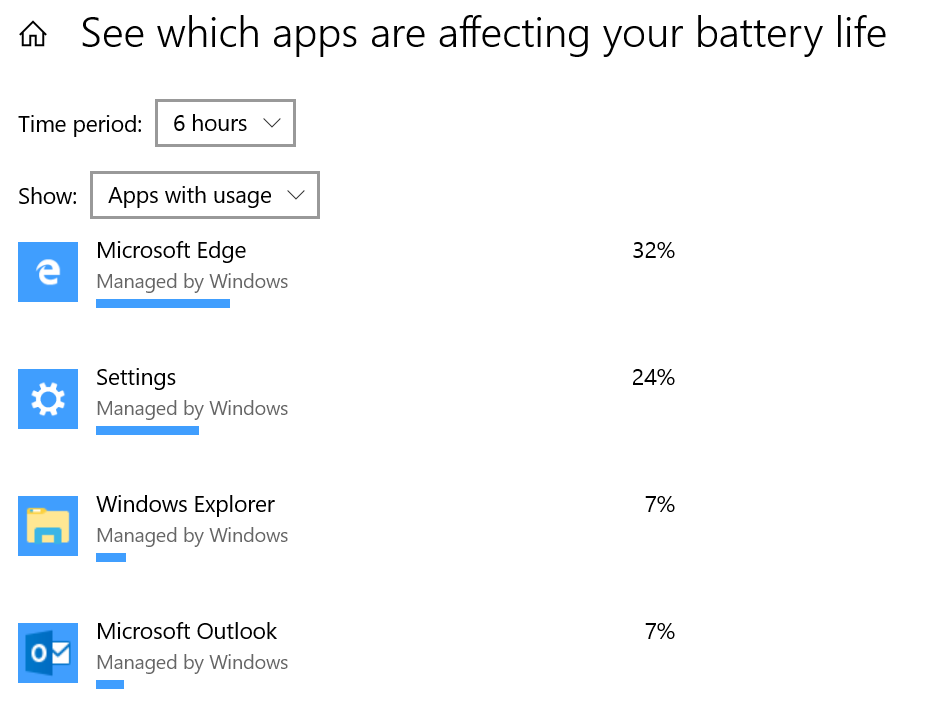
1. Unplug the power cord from the device.
2. From the start menu, type: **Battery Saver Settings**
3. Select *Battery Saver* in System settings and notice how:
   1. Battery Saver is configured to turn on automatically when the battery falls below 20%. This setting is adjustable.
   2. You can manually turn on and turn off the battery saver by toggling the on/off switch.
   3. By default, screen brightness will be reduced by 30% when battery saver is enabled.
   4. Learn more about Battery Saver: <https://aka.ms/windowsbatterysaver>
   5. There is an option to click to view battery usage by app.



1. Plug the power cord back into the device.

## Step 2 – View Battery Usage By App

1. From the Battery Saver Overview window, click on **Battery usage by app** and notice how:
   1. The UI breaks down battery usage by application. The list includes Win32 and UWP user mode applications and some Windows components but excludes services and background processes that may not be familiar to most end users.

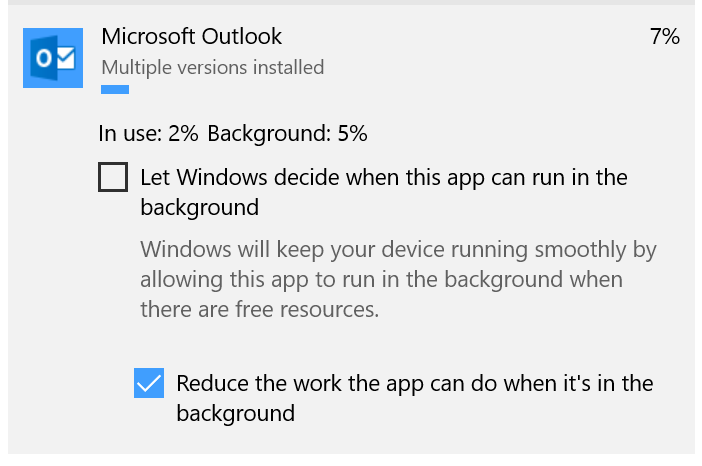


* 1. You can choose to limit the energy data to specific time periods including 24 hours, 6 hours, and 1 week.
  2. You can choose to show:
     1. Apps with usage: Apps that have energy attributed to them.
     2. All apps: All UWP apps including those that have not had energy attributed to them.
     3. Always allowed apps: Apps that are configured by the user or IT department to “Always allowed to run” in the background.

1. Clicking on a Win32 app such as Microsoft Power Point provides the user with the following information and options:

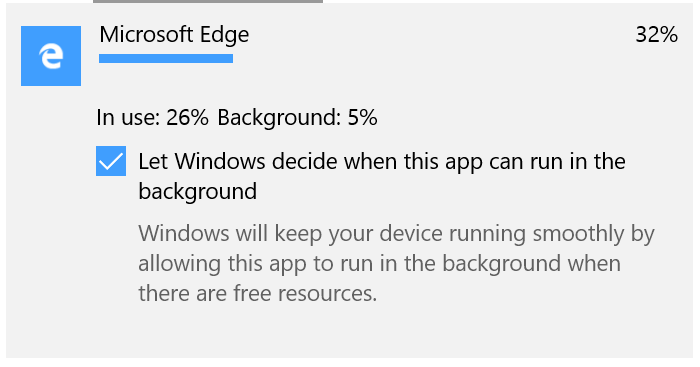


* + 1. Unchecking the “Let Windows decide when this app can run in the background” provides the user with the following option:

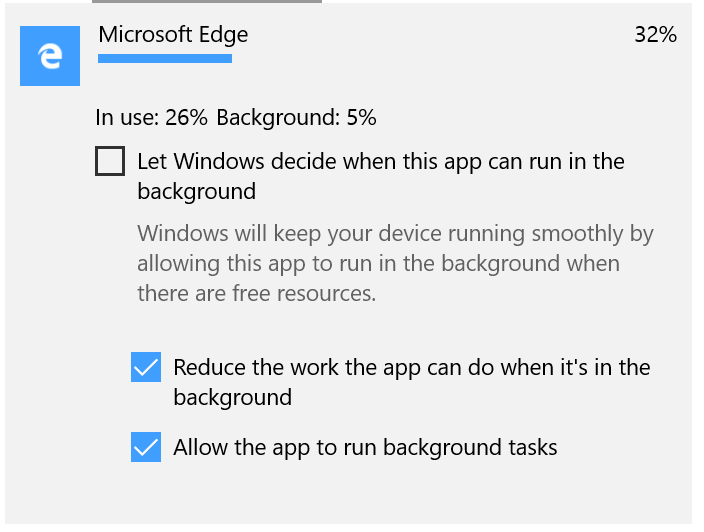


* + 1. When the option labeled “Reduce the work the app can do when it’s in the background” is checked, this app can be power throttled to save power when it’s not in the foreground. Unchecking this option will opt the app out of being power throttled. Additional details on power throttling are provided in the Performance Power Slider section of this lab.

1. Clicking on a UWP app such as Microsoft Edge provides the user with the following information and options:



* + 1. Unchecking the “Let Windows decide when this app can run in the background” provides the user with the following options:



* + - 1. When the option labeled “Reduce the work the app can do when it’s in the background” is checked, this app can be power throttled to save power when it’s not in the foreground. Unchecking this option will opt the app out of being power throttled.

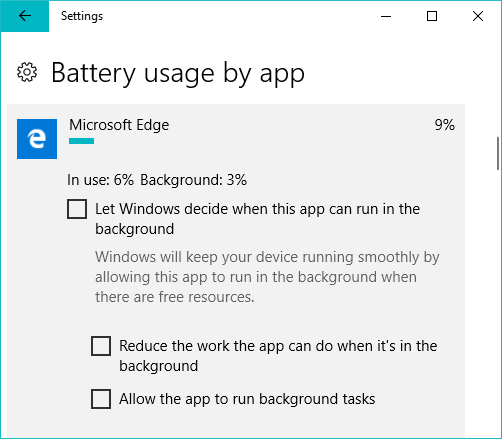
When the option “Allow the app to run background tasks” is checked, UWP apps such as the Mail app are allowed to run and sync email when the device is in Modern Standby.

# Exercise 2A – Perf Power Slider: Opt out of Power Throttling

**Estimated Time**: 5 minutes  
**Level**: 100  
  
In this exercise, you will learn how users can opt out of power throttling. Users can opt out specific apps using the **battery usage by app** UI. Users can opt all apps out of power throttling by moving the performance power slider to the Best Performance position. Power users can use the details tab in Task Manager to confirm that specific apps are not power throttled. Power throttling is only available on DC power in the following slider positions: Battery Saver, Better Battery, and Better Performance.

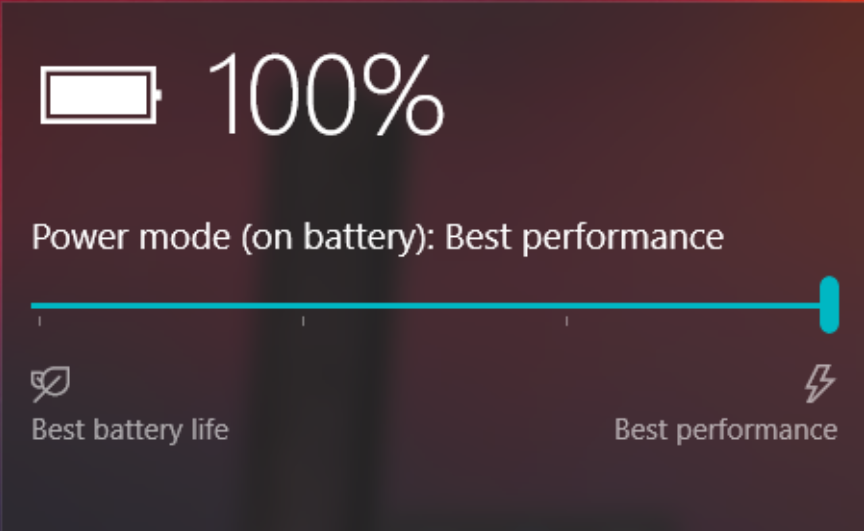
Note: The Better Battery slider position may appear as Recommended position if the user upgraded from Creators Update.

## Step 1 – Opt specific apps out of power throttling in Battery Usage by app UI

1. Verify device is unplugged from AC power.
2. From the start menu, type “B*attery saver*”, then select **Battery Saver.**
3. Click on **Battery usage by app** then select the app that you’d like to opt out of power throttling.
4. Uncheck “Let Windows decide when this app can run in the background”.
5. Uncheck "Reduce the work the app can do when it’s in the background". In the screenshot below, Edge will no longer be power throttled when running in the background on DC power.  
   

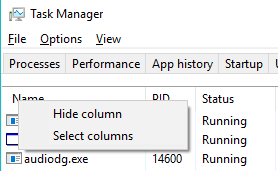
## Step 2 – Opt all apps out of power throttling using the slider

1. Verify device is unplugged from AC power.
2. Click the battery icon
3. Move the Performance power slider position to the best performance position.

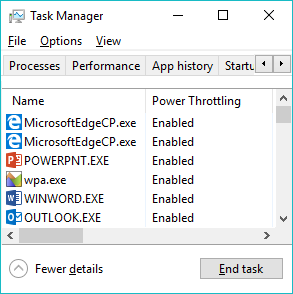


## Step 3 – Verify apps are power throttled using Task Manager

1. Verify device is unplugged from AC power.
2. Press **CTRL+SHIFT+ESC** to launch the task manager
3. Click on the **Details** tab
4. **Right click** on any column header and click **Select Columns**



1. Scroll down and select **Power Throttling**, then press **OK**.
2. Take note as you move an app from the foreground to the background, it changes from disabled to enabled.
3. Understand the meaning of the values:
   1. Disabled = app is not being power throttled
   2. Enabled = app is being power throttled
4. Notice how all apps shown in the following screenshot are being power throttled since task manager is in the foreground.



**Notes**

* Common causes of all apps being disabled:
  + Device is on AC power.
  + Slider is in the Best Performance position.
  + Chipset may not support power throttling.

# Exercise 2B –Perf Power Slider: Viewing PPM settings in WPA

**Estimated Time**: 5-10 minutes  
**Level**: 200  
  
In this exercise, you will learn how to look up the PPM settings that are applied to each position on the Performance Power Slider.

## Step 1 – Collect power logs for 2 Slider Positions

1. Unplug the device from AC power.
2. Move the Performance Power Slider to the **Better Battery** position
   1. Note: on devices that were upgraded from RS2 or customized by an OEM, this position may appear as **Recommended**.
3. From the start menu, type “*Command Prompt*”, right click and select “*Run as administrator*” then run the following commands:

*wpr -cancel*

*wpr -start power -filemode*

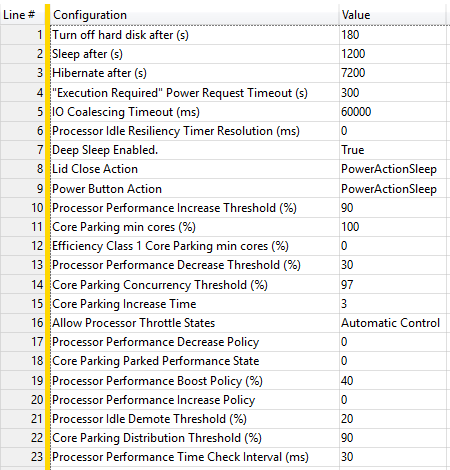
1. Wait 5 seconds then run the following command:

*wpr -stop BetterBattery.etl*

1. Repeat steps 2 - 4, however, in step 2, move the slider to the **Best Performance** position and in step 4, name the ETL file **BestPerformance.etl**.
2. Plug the device into AC power.

## Step 2 – View Power Logs in WPA

1. Launch WPA by typing “*WPA*” from the start menu.
2. Click on **File**, then click **Open** and navigate to the **BetterBattery.etl**
3. Click on the **Trace** file menu, then **System Configuration**
4. Click on **Power Settings**
5. View the long list of more than 50 PPM setting names and values.



1. Repeat steps 1-5, however, in step 2, open the **BestPerformance.etl** trace and compare the values with the **Better Battery** position.

**Notes**

* Microsoft does not expect OEM/ODM partners to have to modify any of these PPM settings.
* Microsoft recommends that OEMs interested in tuning these PPM settings for other usages contact their silicon vendor for guidance.

# Exercise 2C –Perf Power Slider: Viewing Overlay Settings using PowerCfg

**Estimated Time**: 5-10 minutes  
**Level**: 200  
  
In this exercise, you will learn how to look up the overlay settings that are applied to each position on the Performance Power Slider. Windows includes a set of default Energy Performance Preference (EPP) values which Intel chipsets with Speed Shift technology use for favoring performance (EPP 0) or power (EPP 100). Intel Speed Shift is required to take advantage of EPP so if the settings exist on a device that does not support EPP (or another HWP interface) then they do not take effect.

These EPP values can be customized by OEMs using Windows provisioning and queried using PowerCfg commands found in this exercise. In addition to EPP settings, OEMs can define their own custom overlay settings in their firmware for each slider position. Learn more: <https://aka.ms/perf-power-slider> (Search for AddPowerSettingDirective).

## Step 1 – Run PowerCfg Commands to Query PPM and EPP Settings

1. From the start menu, type “*Command Prompt*”, right click and select “*Run as administrator*”.
2. Run the following commands to look up the overlay PPM and EPP settings for each slider position:

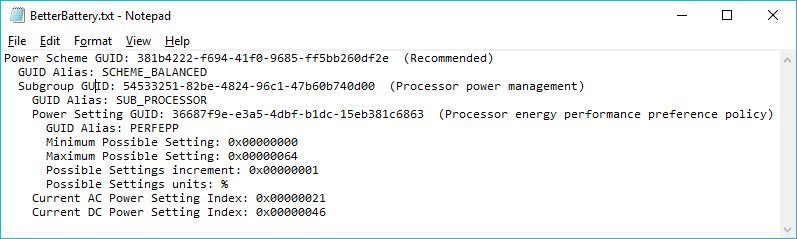
Powercfg /QH OVERLAY\_SCHEME\_MIN > BetterBattery.txt

Powercfg /QH SCHEME\_BALANCED SUB\_PROCESSOR PERFEPP > BetterPerformance.txt

Powercfg /QH OVERLAY\_SCHEME\_MAX > BestPerformance.txt

## Step 2 – View PPM/EPP Settings in notepad and convert values to decimal

1. Open all three text logs to view the EPP values for AC and DC power.
2. Let’s take a look at sample output for the **Better Battery** overlay:



1. Notice the EPP value for AC is **0x00000021** and DC is **0x00000046**.
2. Follow these steps to convert these hex values to decimal:
   1. From the start menu, type *“Calculator”* and click **Calculator**.
   2. Select **“Programmer”** from the menu.
   3. As shown in the screenshot below, click on HEX and type *“46”*, then take note of the DEC value. Repeat for
      1. AC: 0x00000021 = EPP 33
      2. DC: 0x00000046 = EPP 70

**Notes**

* Intel Speed Shift technology is required for EPP values to take effect.
* Use official Windows Provisioning packages from your silicon vendor to define EPP values on official OEM images.
* Learn more:
  + Power Settings: <https://aka.ms/Config-Power-Settings>
  + Power Slider: <https://aka.ms/Perf-Power-Slider>
  + PPM Options: <https://aka.ms/PPM-Options>
  + PerfEPP: <https://aka.ms/PerfEPP>
  + Power Throttling Blog: <https://aka.ms/powerthrottling>
* The following PowerCfg commands can be used to modify the EPP values and are only intended for experimentation on a test device.
* **Better Battery***powercfg /setdcvalueindex OVERLAY\_SCHEME\_MIN SUB\_PROCESSOR PERFEPP 70*
* **Better Performance***powercfg /setdcvalueindex SCHEME\_BALANCED SUB\_PROCESSOR PERFEPP 50  
  powercfg /setacvalueindex SCHEME\_BALANCED SUB\_PROCESSOR PERFEPP 33*
* **Best Performance**  
  *powercfg /setdcvalueindex OVERLAY\_SCHEME\_MAX SUB\_PROCESSOR PERFEPP 33  
  powercfg /setacvalueindex OVERLAY\_SCHEME\_MAX SUB\_PROCESSOR PERFEPP 25*

# Exercise 2D – Perf Power Slider: Adding custom OEM settings

**Estimated Time**: 5-10 minutes    
**Level**: 300  
    
In this exercise, you will learn how to add custom settings from your driver to the Performance Power slider. The AddPowerSetting directive <https://aka.ms/AddPowerSetting> allows OEMs/SVs to configure their custom settings and plug into the window power and policy store. We will use this directive to configure custom settings for the slider.

**Note**: Only settings which can affect perceived performance differences should be customized across slider modes. Each slider mode should be thought of as a “lite” power plan, which only contains settings that impact performance. Examples of performance settings that are good candidates for the performance power slider: fan speed, thermals, and GPU settings. Examples of settings that are not intended to be controlled by the performance power slider include: keyboard backlight and sleep settings.

## Step 1 – Run PowerCfg Commands to Query CURRENT Overlay Settings

1. From the start menu, type “*Command Prompt*”, right click and select “*Run as administrator*”.
2. Run the following commands to look up the overlay PPM and EPP settings for each slider position:

Powercfg /QH OVERLAY\_SCHEME\_MIN > BetterBattery.txt

Powercfg /QH SCHEME\_BALANCED SUB\_PROCESSOR PERFEPP > BetterPerformance.txt

Powercfg /QH OVERLAY\_SCHEME\_MAX > BestPerformance.txt

## Step 2 – Add Custom Power setting to your Driver’s INF

In this section we’ll create a custom setting for modifying the fan speed of a device based on performance power slider positions. The fan speed setting created below has three possible setting values:

1. Low: We will configure the better battery position to be Low (minimum fan speed)
2. Medium: Better performance to be Medium (medium fan speed) on DC and High on AC
3. High: Best Performance to be High (maximum fan speed).

The idea is to provide the user with improved performance when moving from Better Battery to Best Performance. These settings are just an example for the dummy fan speed power settings, you might want to configure the behavior of your devices differently but please adhere to these general design principles of the slider.

Note: this lab does not contain sample code or binaries for the fan speed driver/firmware. The purpose of this exercise is to introduce you to the general concept on how this works.

// Within a DDinstall or ClassInstall23 section

AddPowerSetting=LCDDim

[LCDDim]

SubGroup = {oem subgroup} ; TODO: Add your custom subgroup

Setting = {settings GUID}, "Fan Speed", "Controls the speed of the Fan" ; TODO: Add your custom setting GUID

Value = 0, "Low", "Minimum Fan Speed", %FLG\_ADDREG\_TYPE\_DWORD%, 0x50

Value = 1, "Medium", "Medium Fan Speed", %FLG\_ADDREG\_TYPE\_DWORD%, 0x75

Value = 2, "High", "Maximum Fan Speed", %FLG\_ADDREG\_TYPE\_DWORD%, 0x100

; Power slider overlay defaults 

Default = %GUID\_BETTER\_BATTERY%, %AC%, 0 ; Minimum fan speed for Better Battery 

Default = %GUID\_BETTER\_BATTERY%, %DC%, 0

Default = %GUID\_BEST\_PERFORMANCE%, %AC%, 2 ; Maximum fan speed for Best Performance

Default = %GUID\_BEST\_PERFORMANCE%, %DC%, 2

; Power scheme defaults 

Default = %GUID\_TYPICAL\_POWER\_SAVINGS%, %AC%, 2 ; Balanced settings for the balanced scheme, 

Default = %GUID\_TYPICAL\_POWER\_SAVINGS%, %DC%, 1 ; aka the ‘Better Performance’ position

Default = %GUID\_MAX\_POWER\_SAVINGS%, %AC%, 0 

Default = %GUID\_MAX\_POWER\_SAVINGS%, %DC%, 0

Default = %GUID\_MIN\_POWER\_SAVINGS%, %AC%, 2 

Default = %GUID\_MIN\_POWER\_SAVINGS%, %DC%, 2

[Strings]

GUID\_MAX\_POWER\_SAVINGS = {a1841308-3541-4fab-bc81-f71556f20b4a}

GUID\_TYPICAL\_POWER\_SAVINGS = {381b4222-f694-41f0-9685-ff5bb260df2e}

GUID\_MIN\_POWER\_SAVINGS = {8c5e7fda-e8bf-4a96-9a85-a6e23a8c635c}

GUID\_BETTER\_BATTERY = {961CC777-2547-4F9D-8174-7D86181b8A7A}

GUID\_BEST\_PERFORMANCE = {DED574B5-45A0-4F42-8737-46345C09C238}

AC = 0

DC = 1

FLG\_ADDREG\_TYPE\_DWORD = 0x00010001

**Note**: Deploying and installing driver is beyond the scope of this exercise. For more information on creating custom power settings from your driver please reference the AddPowerSetting MSDN documentation referenced above.

## Step 3 –  Verify your setting was added via powercfg

After you installed a driver that follows the guidance above, you should verify the custom settings across all slider positions. For example, below we’ll take a look at the fan speed settings under the Best Performance slider position.

Run the following command to verify the Fan speed setting exists:

**powercfg /qh overlay\_scheme\_max > max\_overlay.txt**

  Subgroup GUID: {Custom OEM GUID} (OEM Settings)

    Power Setting GUID: SETTING\_GUID  (Fan Speed)

      Possible Setting Index: 000

      Possible Setting Friendly Name: Low

      Possible Setting Index: 001

      Possible Setting Friendly Name: Medium

      Possible Setting Index: 002

      Possible Setting Friendly Name: High

    Current AC Power Setting Index: 0x00000002

    Current DC Power Setting Index: 0x00000002

**Note:** This exercise covered the creation and initialization of a custom setting. After you’ve created and specified defaults for a custom you should register for the PoRegisterPowerSettingCallback <https://aka.ms/PoRegisterPowerSettingCallback> (kernel-node) or PowerSettingRegisterNotification <https://aka.ms/PowerSettingRegisterNotification> (user-mode) to register a power-setting callback routine to receive notifications of changes in the specified power setting.

# Exercise 3A – Accessing Energy Estimation Engine (E3) data using PowerCfg

**Estimated Time**: 10 minutes  
**Level**: 200

In this exercise, you will use PowerCfg to output the Energy Estimation Engine data which is stored in a local SRUM database into a CSV or XML formatted file.

## Step 1 – View PowerCfg /SRUMUTIL usage

1. *From the start menu, type Command Prompt, right click and select “Run as administrator”*
2. *Run “powercfg /SRUMUTIL /?”*
3. This command provides the usage instructions (also shown below).

POWERCFG /SRUMUTIL [/OUTPUT <FILENAME>] [/XML | /CSV]

Description: Dumps entire SRUM DB for energy estimation provider in XML or CSV format

Parameter List:

/OUTPUT <FILENAME> Specify the path and filename to store the SRUM data in CSV or XML file.

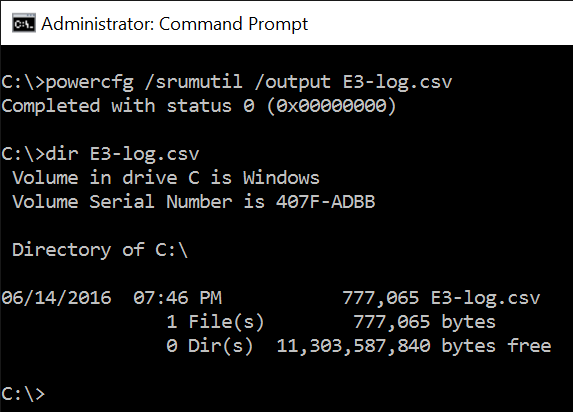
/XML Format the report file as XML.

/CSV Format the report file as CSV.

Examples: POWERCFG /SRUMUTIL /OUTPUT "srumdbout.xml" /XML

## Step 2 – Output SRUM data to CSV format

1. Using the same elevated command prompt, output the Energy Estimation Engine data to a CSV file by running the following command: “**powercfg /srumutil /output E3-log.csv**”



1. Generate an Energy Estimation Engine log formatted in XML by running the following command: “**powercfg /srumutil /xml /output E3-log.xml**”

# Exercise 3B – Visualizing Energy Estimation Engine data in PowerView

**Estimated Time**: 15 minutes  
**Level**: 300

In this exercise, you will learn how to visualize energy data collected in the previous exercise using PowerView. A variation of this exercise can be viewed in this online training video: <https://aka.ms/DefragTools-E3> Before jumping into pivot tables, we’ll review the meaning of the columns in the E3 logs.

## Step 1 – Understanding the meaning of E3 Logs

1. The E3 SRUMUtil data provides the estimated energy usage per application/process per component. The unit of the reported energy is in milli-joules, which is equal to (seconds \* milli-watts). Each log includes a large number of fields.

For a definition of all E3 log fields, refer to Annex II

1. A common question is what’s the difference between software estimation and hardware measurements? The following table calls out the approximate difference:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| E3 Configuration | Approximate accuracy per E3 power model | | | |
| **CPU** | **Disk** | **Display** | **Network** |
| Software Estimation \* | 89% \*\*\* | <70% | <70% | <70% |
| Hardware Measurement \*\* | >98% | >98% | >98% | >98% |

\* Actual accuracy of the software estimation code path varies depending on how different the hardware power characteristics are from a device Microsoft used to generate the default inbox power profiles.

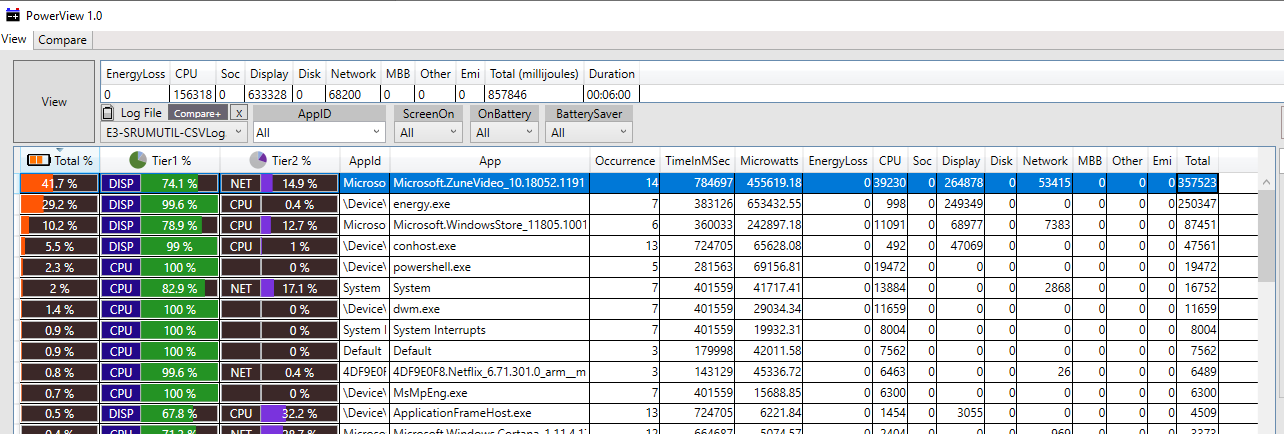
\*\* Hardware based power monitoring chips can be included in pre-production and retail devices.

\*\*\* This varies from processor to processor, but the 89% value is based on energy data reported by Intel MSRs on recent Intel chipsets.

Note: the accuracy data above is specific to screen-on scenarios.

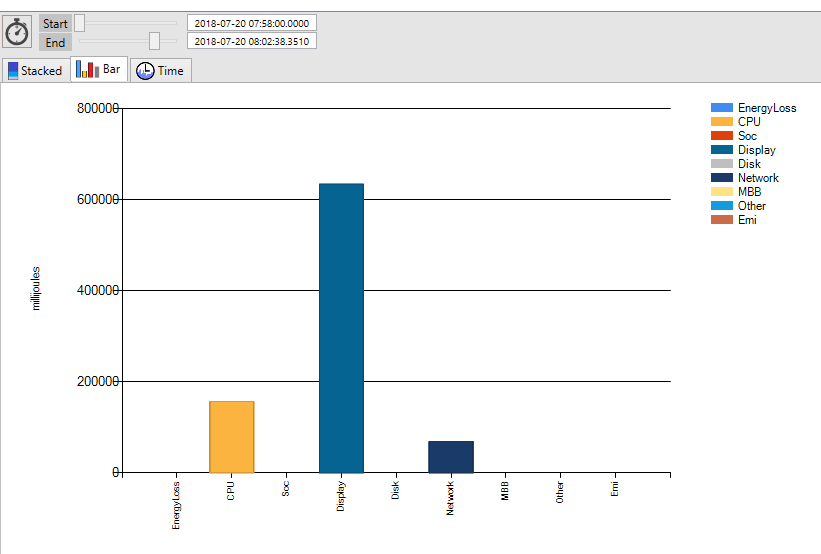
## Step 2 – Listing the top processes/Power Rails Consuming energy

1. Open **PowerView** from the Start menu
2. Open **E3-log.csv** by clicking on the **“View”** button  
   Note: A sample E3 log generated on a device with a hardware power meter is available in "C:\BatteryLife\EnergyEstimation\srumutil.csv"
3. You now see the list of all processes sorted by total energy consumed
   1. Total % represents the percentage of total system energy consumption attributed to the process (e.g. 41.7 % of the total energy consumption was due to ZuneVideo)
   2. Tier 1 % represents the percentage of process energy consumption attributed to the top power rail (e.g. 74.1% of the energy consumed by ZuneVideo was due to the display)
   3. Tier 2 % represents the percentage of process energy consumption attributed to the second top power rail (e.g. 14.9% of the energy consumed by ZuneVideo was due to the network)



## Step 3 – Visualizing Energy consumption per E3 Power Model

1. On the right side of the PowerView UI, you can see the distribution of total energy consumed and segmented by power rails.
2. Different views can be selected by clicking on Stacked, Bar or Time.
3. In this example, you can see that Display, CPU and Network are the top power rails where energy is consumed.



## Step 4 – Clearing the E3 SRUMUTIL database for test purposes

1. Partners interested in using the E3 data to track energy consumption from build to build or to compare setting A vs. B can clear the E3 data in order to start an energy efficiency workload in a controlled test environment. The battery life assessments in the ADK use this technique to collect E3 logs.
2. Launch an elevated command prompt and type the following commands:  
   *sc stop dps*  
   *move /y %windir%\system32\sru\srudb.dat srudb.dat.bak*  
   *sc start dps*

# Exercise 3C – Collecting & Visualizing Energy Estimation ETW events in WPA

**Estimated Tim**e: 15 minutes  
**Level**: 300

The following ETW provider can also be used to collect Energy Estimation Engine data:

Microsoft-Windows-Energy-Estimation-Engine

In contrast to PowerCfg /SRUMUTIL which queries the SRUM database for historical information, ETW providers log current energy information. By default, the Energy Estimation Engine ETW provider collects Energy Estimation Engine data once per minute.

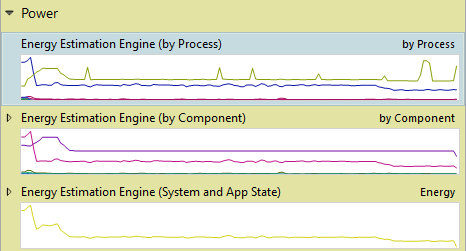
## Step 1 - Collect an ETW trace using WPR

1. From the start menu, type “*Command Prompt*”, right click and select “*Run as administrator*”.
2. Run “*cd /d C:\BatteryLife\EnergyEstimation*”
3. *wpr -cancel*
   1. *Note: Ignore any errors that are returned by this command*
4. *wpr -start EnergyEstimationEngine.wprp -filemode*
5. Wait at least 5 minutes. You can use this time to look at the next exercise.
6. *wpr -stop E3.etl*

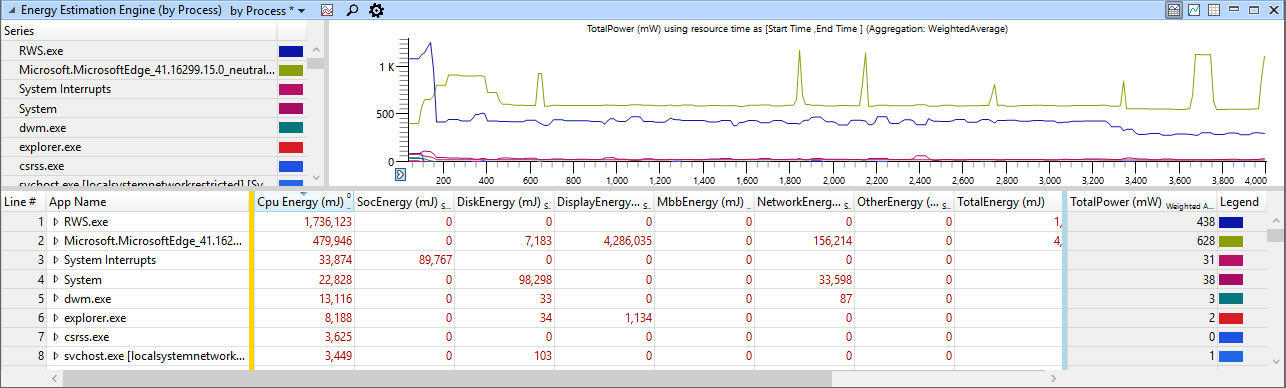
## Step 2 - Visualize the ETW trace in WPA

1. Open the E3.etl file using WPA (Windows Performance Analyzer) by running  
   *wpa E3.etl*

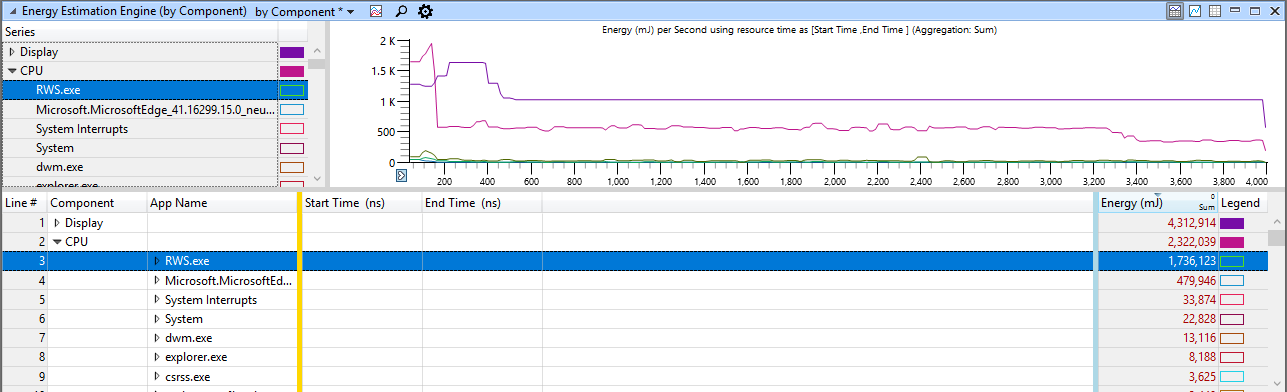
*Note: A sample E3 ETW log can be found here: "C:\BatteryLife\EnergyEstimation\E3.etl"*

1. Expand the Power node in Graph Explorer.
2. Notice how the energy views are broken down by process, component, and system & app state.  
   
3. Double click on each of the Energy Estimation Engine views to display the analysis charts and tables. Expand the various nodes, filter the views to specific components or processes, and experiment with the views.

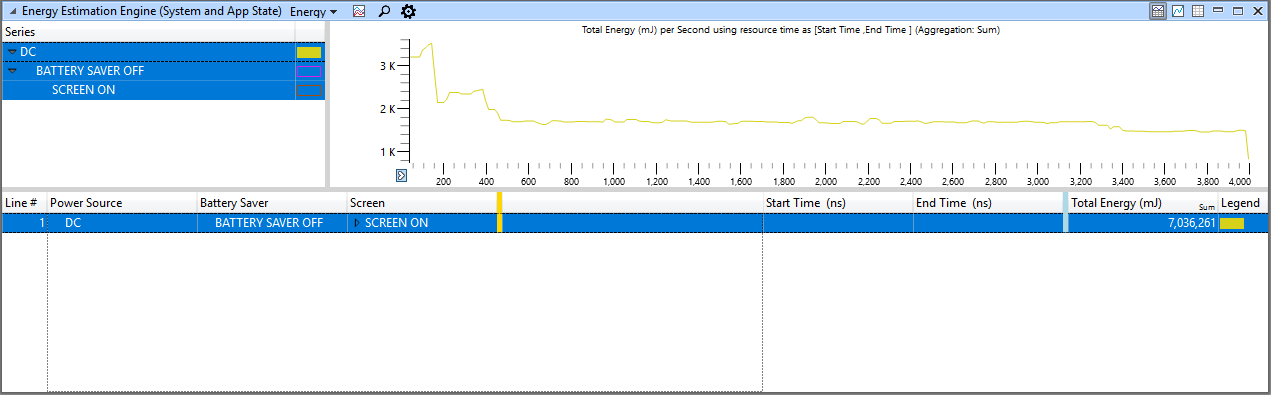
Energy Estimation By Process:



Energy Estimation By Component:



Energy Estimation By System and App State:



## Step 3 - Understanding the meaning of the ETW event property payloads

Please refer to the column definitions in the “Visualizing Energy Estimation Engine data in Excel” exercise.

# Exercise 4 – Battery Life Assessments

**Estimated Time**: 30 min to several hours

Actual test execution time depends on how long the assessment is configured to run for and the drain rate of the device under test. Tip: when evaluating ADK jobs, you can increase the screen brightness to 100% to increase the drain rate and decrease the test execution time.

Level: 200

In this exercise, we’ll execute Energy Efficiency assessments (also referred to as workloads) in the Assessment and Deployment Kit (ADK), view the high-level metrics in the Windows Assessment Console (WAC), and look at the location of the ETW & E3 logs that are generated by the assessment. The high-level metric that we’ll focus on while viewing results in the WAC is “Projected Time to Shutdown”. Details on how to analyze the ETW logs can be found in a subsequent exercise in this lab. Details on how to visualize the E3 data in Excel are described in a previous exercise within this lab.

Note: Assessments are supported on Windows on Snapdragon, however, the assessments must be packaged up into jobs on an X86/AMD64 device then executed from the command line. Microsoft recommends installing the Windows ADK Insider Preview build 17040 or greater when running assessments on Windows on Snapdragon.

Before continuing in this exercise, please ensure each of the following required pre-requisites are met:

|  |  |
| --- | --- |
| **Assessment** | **Required pre-requisites** |
| Local Full Screen Video playback (FSVP) in Movies & TV app – 1080p Soccer clip | * The user logged into Windows must be a local (non-domain) user. This assessment does not work when logged in as a domain user. |
| Streaming FSVP (premium content) in Movies & TV app – Halo 2 clip | * The user logged into Windows must be a local (non-domain) user. This assessment does not work when logged in as a domain user. * Sign into the Microsoft Store and the Movies & TV app using a valid Microsoft Account. This is required in order to add the Halo 2 clip to a user’s library. * Set the region in Windows to United States. cid:image002.jpg@01D35E67.98765290 |
| Netflix streaming | * Install the Netflix UWP app from the Microsoft Store. * Log into Netflix using a valid Netflix account that supports streaming. * If multiple users exist for the account, select a user. |
| Office Productivity | * Install Office 2016 (Word, Excel, Power Point, Outlook, and OneNote). * Sign into Outlook and OneNote using a valid email account. * **IMPORTANT NOTE**: This assessment will respond (reply all) to the last email that was sent to the email address, so it is highly recommended that a test email account be used for this assessment. |

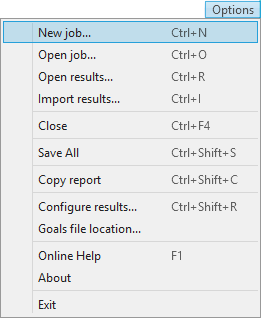
## Step 1 – Prepare the device for the assessments

On the device under test, follow these steps to prepare the system for the assessment:

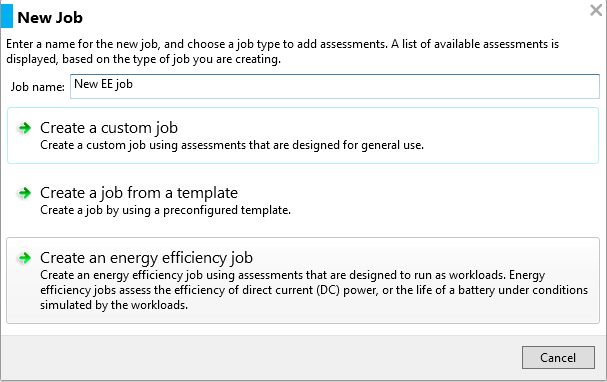
1. Install latest version of **Windows, Windows Updates, and OEM updates**.
2. **Connect the device to the internet** by connecting to a Wi-Fi access point or mobile broadband connection.
3. **Activate Windows**.
   1. If Windows is not activated, then the video playback workloads will not utilize all of the power saving features in Windows due to the “Activate Windows” overlay that is displayed on top of the video.
4. **Complete all pre-requisites** listed in the beginning of this exercise.
5. Verify the device is **logged on as a local admin user**.
6. **Charge the device** to 98% battery life, or the battery life level that you’d like to start the assessment at.
7. Set the **volume level to 50%**, or any consistent volume level**.**
8. If a nit meter is available, **set the screen brightness** to a level that matches 150 nits. If a nit meter is not available, use the default screen brightness for the device.
9. **The date, time, and location** on the test device must be set appropriately in order for the tests to run (PlayReady requirement).

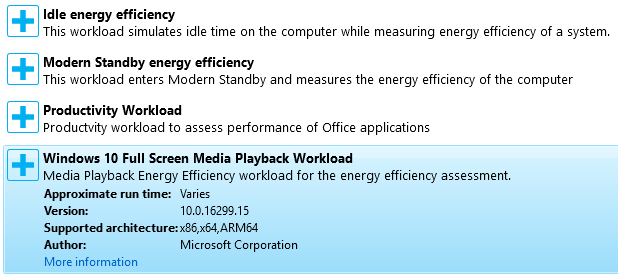
## Step 2 – Execute an Energy Efficiency Assessment

1. Launch the **Windows Assessment Console** (WAC) and follow these steps:
   1. Click **Options**, then **New Job**

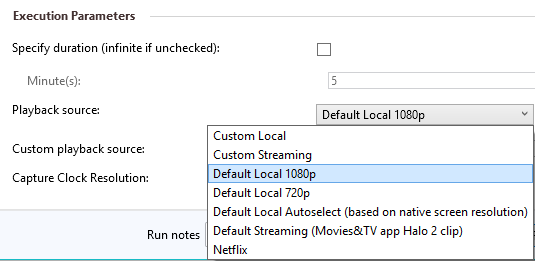


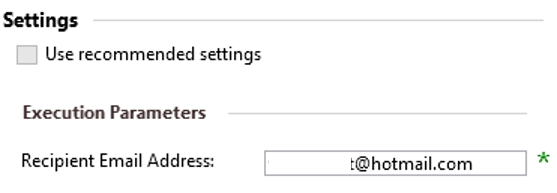
* 1. Select “**Create an energy efficiency job**”

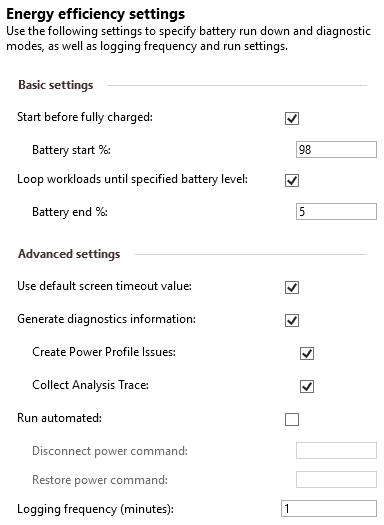


* 1. **Select an energy efficiency workload** to run or package up to be run on another device.  
     

1. Configure the **assessment parameters**.
   1. Full Screen Media Playback (FSVP): Select a scenario to run (Local 1080p in Movies & TV, Stream Halo 2 clip in Movies & TV, Netflix streaming, etc.

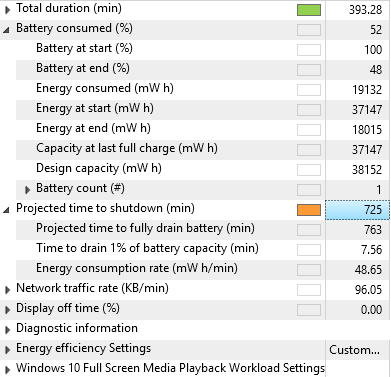


* 1. Office Productivity: Enter the test email address that the automated assessment will send email to.  
     

1. Configure the energy efficiency parameters.   
   
2. At this point you can either run the job or package it up for executing on another device.
   1. **If running on the same device** that the WAC is installed on, **click Run**.
   2. **If packaging up the job, click** **Package** and save the packaged job. If running on a different device using the packaged job, double click on RunJob.cmd in the folder that the job was packaged in.
3. When prompted, unplug the power cord and start the assessment.
4. Allow the assessment to complete. Do not touch the mouse, keyboard, or screen while the assessment is running.
5. **IMPORTANT NOTE**: It is a known issue that the EE workloads take up to 10 minutes to start.

## Step 3 – View EE metrics in the WAC and diagnostic logs in Explorer

1. Once the assessment is completed, launch the Windows Assessment Console on a device that has the ADK installed.
2. Click on Options 🡪 Open Results 🡪 Browse
3. Navigate to the results folder and select the results XML
   1. If you did not run the assessment on your device and would like to view the sample results that are included in this lab, please navigate to: “C:\BatteryLife\ ADK\JobResults\_DESKTOP-GTF5UER\_2017-1124\_1308-53.500\JobResults\_DESKTOP-GTF5UER\_2017-1124\_1308-53.500.xml"
   2. If you ran the assessment on your device directly from the Windows Assessment Console, the results are likely located in your documents folder under a sub folder called Assessment Results. Example path: “C:\Users\jorgen\Documents\Assessment Results\JobResults\_JORGEN1\_2017-1121\_2246-59.545”
4. Take note of the “Projected Time to Shutdown” metric:



1. Navigate to the folder that contains the ETW and E3 logs which can be used for more granular analysis and optimization in tools such as WPA, MXA, and Excel, respectively.
   1. If you did not run the assessment on your device and would like to view the sample results that are included in this lab, please navigate to: C:\BatteryLife\ ADK\JobResults\_DESKTOP-GTF5UER\_2017-1124\_1308-53.500\000\_EnergyEfficiency\results
   2. If you ran the assessment on your device directly from the Windows Assessment Console, the results are likely located in your documents folder under a sub folder called Assessment Results.
      1. Sample path: C:\Users\jorgen\Documents\Assessment Results\JobResults\_JORGEN1\_2017-1121\_2246-59.545\000\_EnergyEfficiency\results

# Exercise 5A – Verify Audio is Offloaded During FSVP

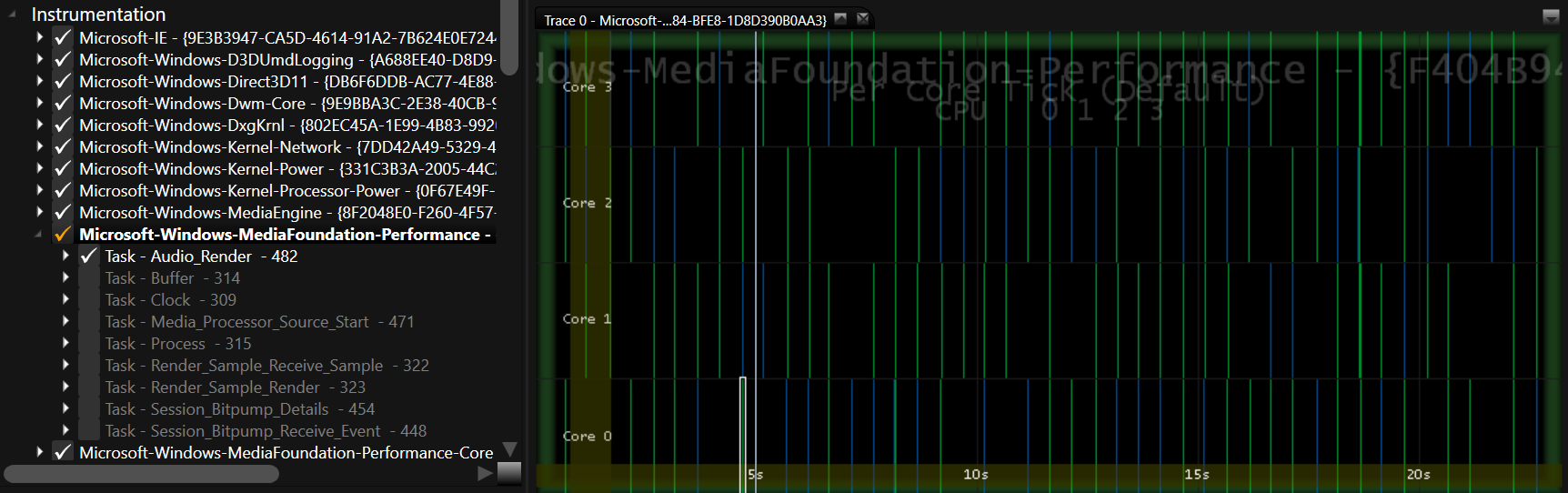
**Estimated Time**: 15 min  
**Level**: 300

In this exercise, we’ll utilize the Media eXperience Analyzer to load the ETW logs generated by ADK FSVP EE assessments previously described in this lab doc to determine whether key power saving features are enabled. More specifically, we’ll look at the ETW logs to verify whether audio is offloaded to hardware. Offloading audio to hardware can improve battery life by up to 14% during FSVP and 50% during Low Power Audio on Modern Standby Devices. To take advantage of audio offload:

* The media player must be media engine based (E.g., Windows Netflix app or Movies & TV app).
* The chipset and audio device must support audio offload.
* The audio driver must support audio offload.

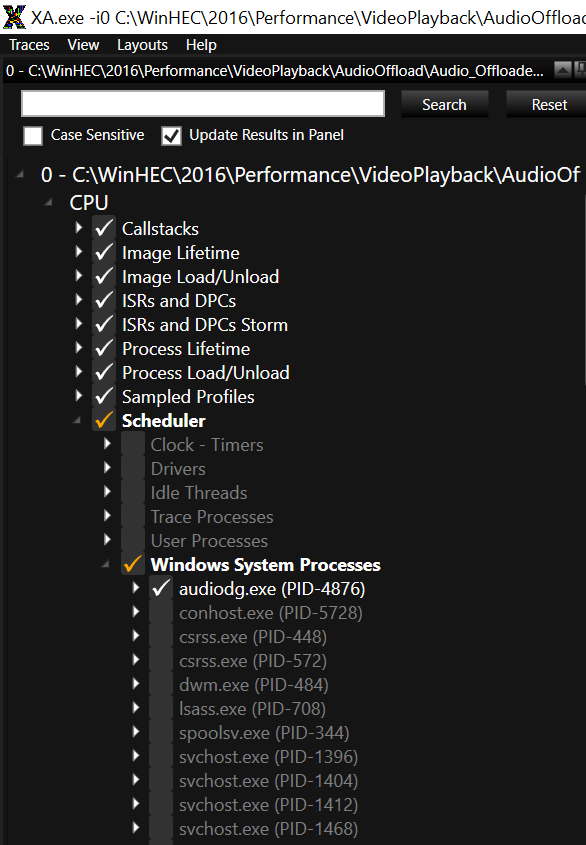
## Step 1 – Load ETW Trace in MXA & Verify Audio is Rendered

1. From the start menu, type Media eXperience Analyzer and open the app
2. When prompted to open file, pass in path to the ETW log generated by the ADK assessment (Local FSVP or Streaming). Examples:
   1. "C:\BatteryLife\VideoPlayback\AudioOffload\Audio\_Offloaded.etl"
   2. "C:\BatteryLife\VideoPlayback\AudioOffload\Audio\_NotOffloaded.etl"
3. Press the **Turn Symbols Off** button to turn off symbol lookup.
4. Once the trace loads, drag and drop the **Microsoft-Windows-MediaFoundation-Performance** provider into a panel.
5. **Deselect** all events in the **Microsoft-Windows-MediaFoundation-Performance** provider by clicking the checkbox next to this dataset two times.
6. Enable the **Task Audio\_Render – 482** events by clicking on this node one time.
7. If the audio render events are fired for the same duration as the content was played when the trace was collected, then we have confirmed that audio was rendered properly when the trace was collected. Per MXA Screenshot 6-1-7, Audio render events in the Microsoft-Windows-MediaFoundation-Performance provider (Task Audio\_Render – 482) are logged throughout the entire trace, this verifies that audio was playing back.

**MXA Screenshot 6-1-7**  


## Step 2 – Determine Whether Audio is Offloaded

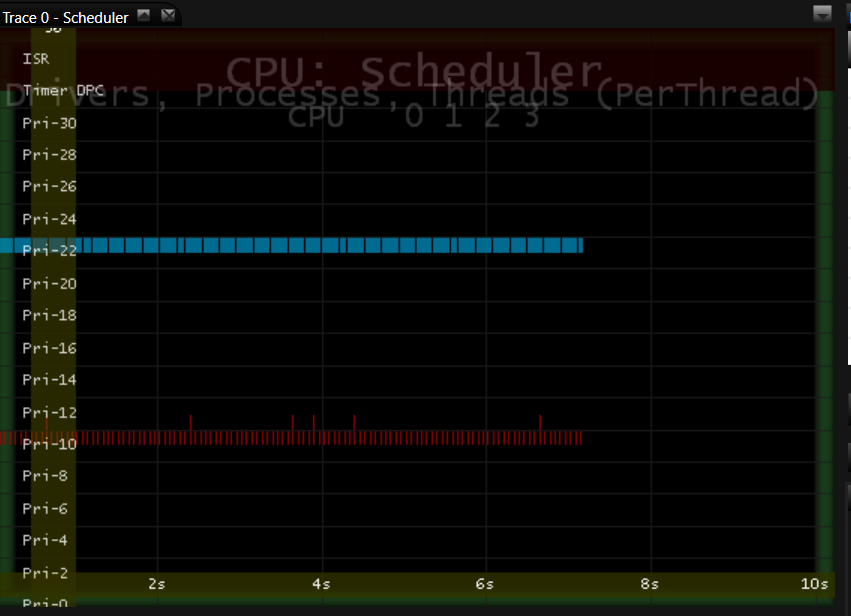
1. Drag and drop the **Scheduler** dataset into a panel.
2. Deselect all events by **clicking** on the **checkbox at the root of the Scheduler dataset twice**.
3. Expand the **Scheduler** dataset and Windows System Processes node.
4. Select the **audiodg.exe**process by clicking the checkbox once.



1. Zoom into a region of the trace that is approximately 1 second

If there is **audiodg.exe** thread activity running at approximately priority 16 or higher every 10ms throughout the entire playback session, then audio is not being offloaded.  See MXA Screenshot 6-2-5 for an example where audio is not offloaded.

**MXA Screenshot 6-2-5** – Audio is not offloaded



1. If there is **audiodg.exe** thread activity running at priority 16 or higher every 500ms or less throughout steady state playback, then audio is being offloaded. See MXA Screenshot 6-2-6 for an example.  On most devices, the frequency of the audiodg.exe threads will likely be less than 500ms.
   1. Note: for low power audio scenarios, the cadence of the audiodg.exe threads when APOs are offloaded is expected to be 1 second.

**MXA Screenshot 6-2-6** – Audio is offloaded

## Step 3 – Review Additional resources on Audio Offload

* <https://aka.ms/AudioOffload>
* <https://aka.ms/AudioOffloadBlog>
* <https://aka.ms/MXA-AudioOffload>

# Exercise 5B – Verify Low Refresh Rate Playback is Enabled During FSVP

**Estimated Time**: 10 min  
**Level**: 300

In this exercise, we’ll utilize the Media eXperience Analyzer to load the ETW logs generated by ADK FSVP EE assessments which are also described in this lab doc to determine whether key power saving features are enabled. More specifically, we’ll look at the ETW logs to verify whether Low Refresh Rate Playback is enabled. To take advantage of low refresh rate, the following criteria must be met:

* The media player must be media engine based (E.g., Windows Netflix app or Movies & TV app).
* The panel must support lower refresh rates.
* The Extended Display Identification Data (EDID) must indicate low refresh rate capability.
* The graphics driver must support low refresh rate playback.

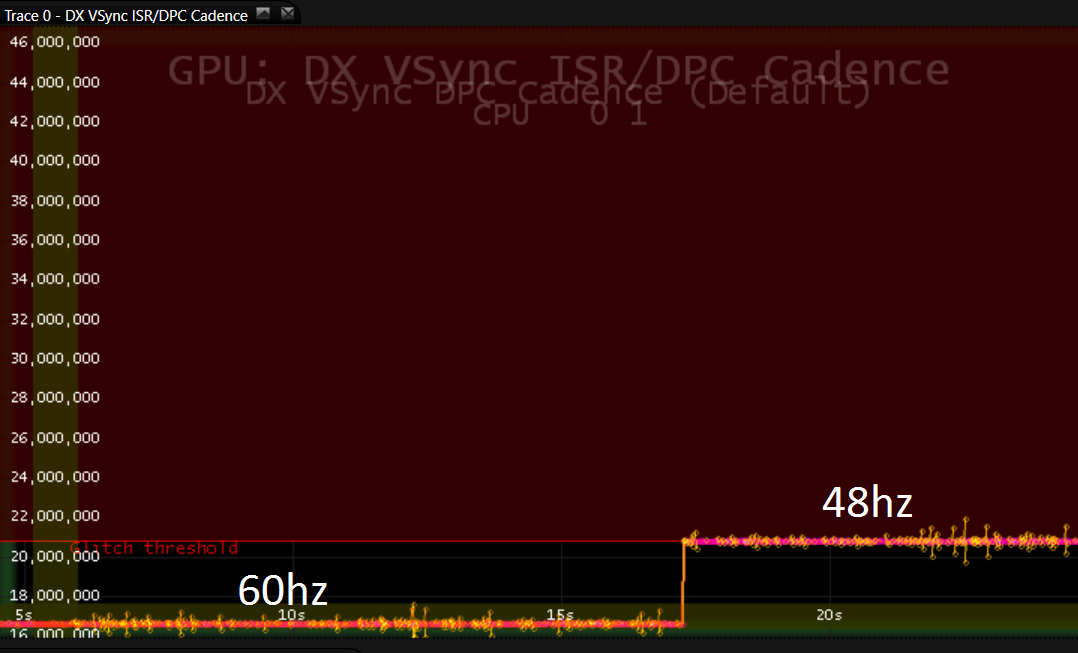
Note: it may take 20-25 seconds after playback starts for 48hz to engage so it’s a good idea to use the steady-state instead of the start-up ETW log in this exercise.

## Step 1 – Load ETW Trace in MXA

1. From the start menu, type Media eXperience Analyzer and open the app
2. When prompted to open file, pass in path to the ETW log generated by the ADK assessment (Local FSVP or Streaming). Example trace: "C:\BatteryLife\VideoPlayback\LowRefreshRate48hz\_MPO\_iFLip\MoviesTVHaloNoProjection.etl"
3. Press the **Turn Symbols Off** button to turn off symbol lookup.

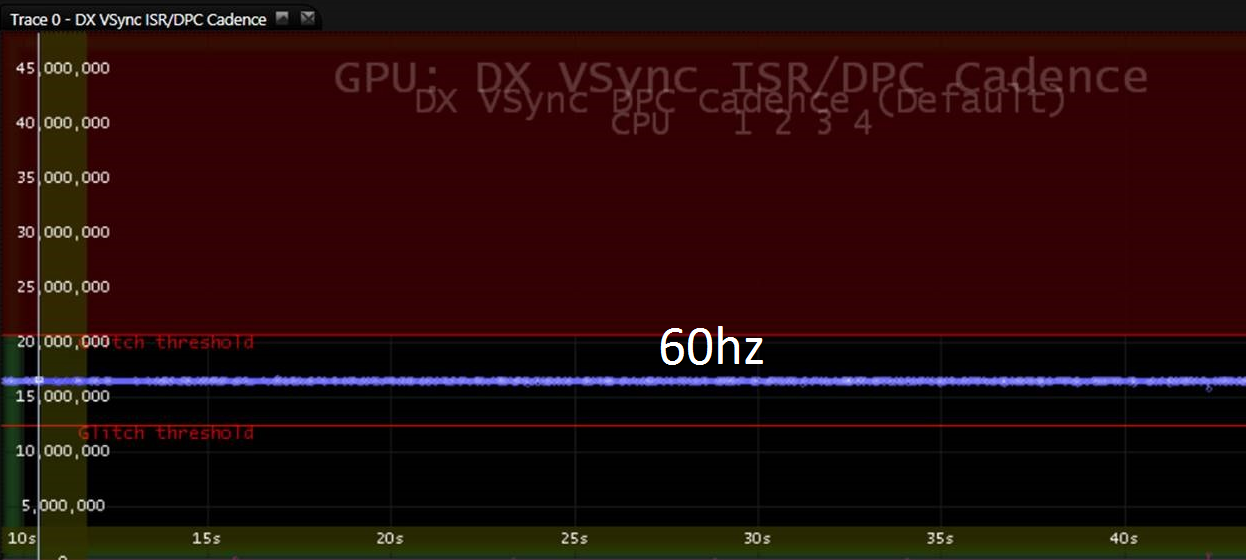
## Step 2 – Determine the Refresh Rate at Various points in the trace

1. **Drag and drop** the **GPU DX VSync ISR/DPC Cadence** dataset into a panel.
2. Zoom into the bottom part of this VSync ISR/DPC Cadence panel during steady state playback
   1. Tip: Use the DMA Operations dataset to help you find steady state playback – look for when DWM activity stops
3. If the VSync interval jumps from 16.6ms to 20.8ms, then 48hz is engaged (see MXA Screenshot #7-2-3).

**MXA Screenshot 7-2-3** – Low Refresh is enabled at approximately 17 seconds into playback  


1. If the VSync interval remains at 16.6ms throughout steady state playback, then 48Hz is NOT engaged (see MXA Screenshot 7-2-4).

**MXA Screenshot 7-2-4 -** Refresh rate remains at 60hz throughout playback (no power savings)



## Step 3 – Review Additional resources on Low Refresh Rate Playback

* Low Refresh Rate: <https://aka.ms/48hzPlayback>
* Power Savings video: Power Savings video: <https://aka.ms/MXA-VideoPower>

# Exercise 5C – Verify MPO and iFlip are Enabled During FSVP

**Estimated Tim**e: 10 min  
**Level**: 300

In this exercise, we’ll utilize the Windows Performance Analyzer to load the ETW logs generated by ADK FSVP EE assessments previously described in this lab doc to determine whether key power saving features are enabled. More specifically, we’ll look at the ETW logs to verify whether MPO and iFlip are enabled. The multi-plane overlay feature, also known as MPO saves power when playing back video. Not all hardware platforms support MPO.

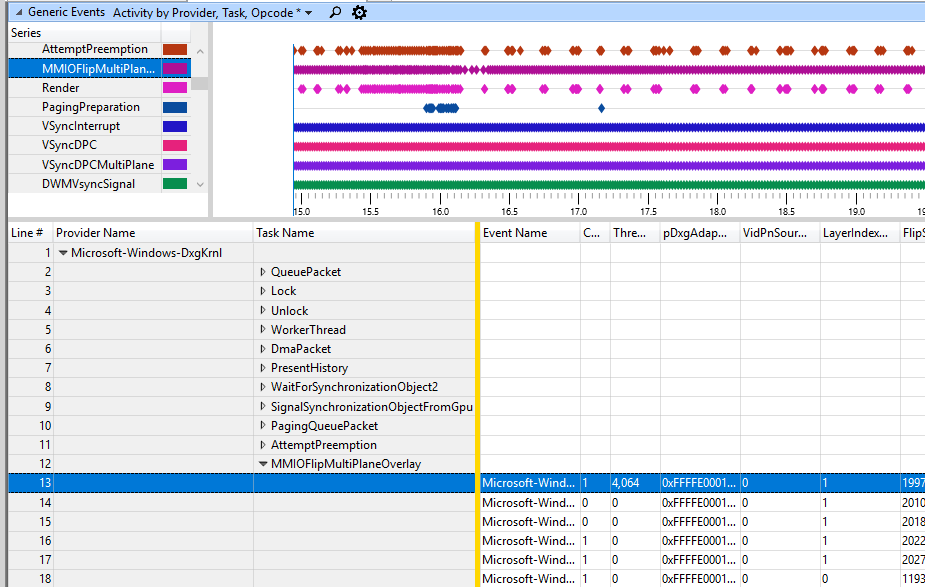
MPO offloads scaling and composition to hardware, reducing memory copies and improving both power and glitch resiliency. Hardware dependency – check with your IHV/OEM to verify whether your hardware supports MPO. Media player must be media engine based (E.g., Windows Netflix app or Movies & TV app).

## Step 1 – Load ETW Trace in WPA

1. From the start menu, type Windows Performance Analyzer and open the app
2. Open the path to the ETW log generated by the ADK assessment (Local FSVP or Streaming). Example: "C:\ BatteryLife\VideoPlayback\MPO\_iFLip\MoviesTVHaloNoProjection.etl"

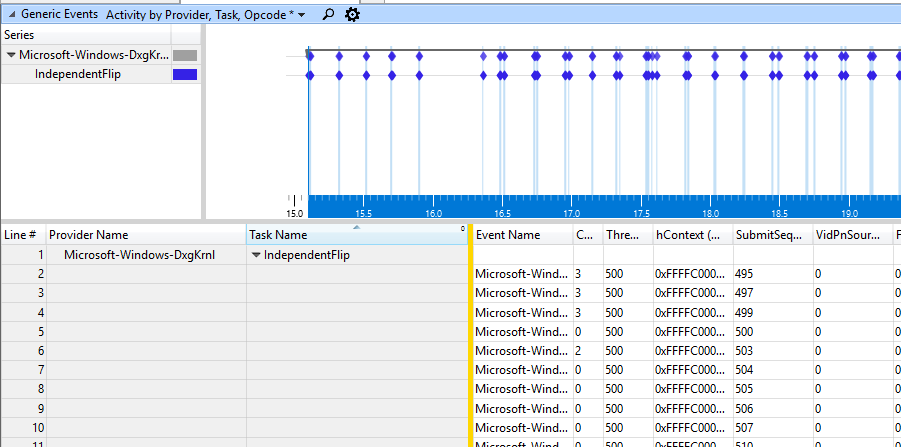
## Step 2 – Determine Whether MPO events exist throughout playback

1. **Drag and drop** the **Generic Events graph** into a panel.
2. Find the **Microsoft-Windows-DxgKrnl** provider
3. Find the **MMIOFlipMultiPlaneOverlay** task.
4. Look at the LayerIndex field of MMIOFlipMultiPlaneOverlay. If there are LayerIndexes higher than 0, then additional planes are used and MPO is enabled



## Step 3 – Determine Whether iFlip events exist throughout playback

1. Find the **IndependentFlip** task.
2. If you notice this event being fired throughout steady state playback, then iFlip is engaged.



## Step 4 – Review Additional resources on MPO & iFlip

* MPO: <https://aka.ms/MPO>
* Power Savings video: <https://aka.ms/MXA-VideoPower>

# Exercise 6A – Modify Power Settings with PowerCfg

**Estimated Time**: 10 min  
**Level**: 200

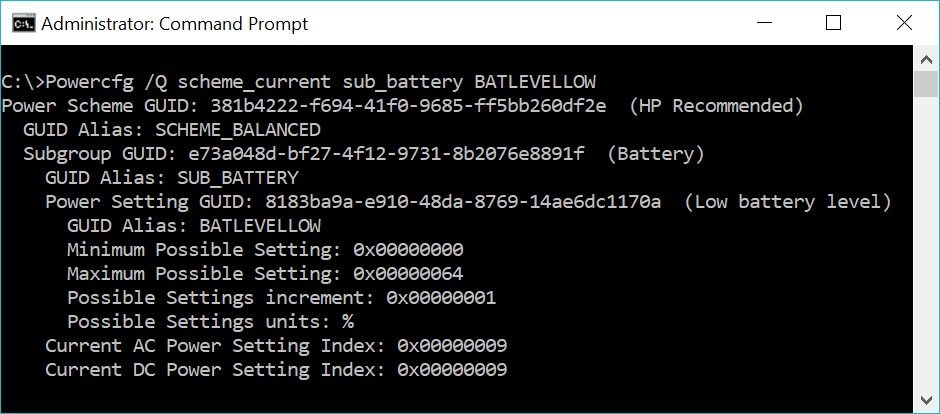
In this exercise, we’ll utilize the inbox component PowerCfg to modify power settings. Please note that using PowerCfg to modify power settings is intended for test and evaluation purposes only. Use Windows provisioning to define custom power settings in OEM images.

## Step 1 – Lookup Current and Set New Critical Battery & Low Battery Values

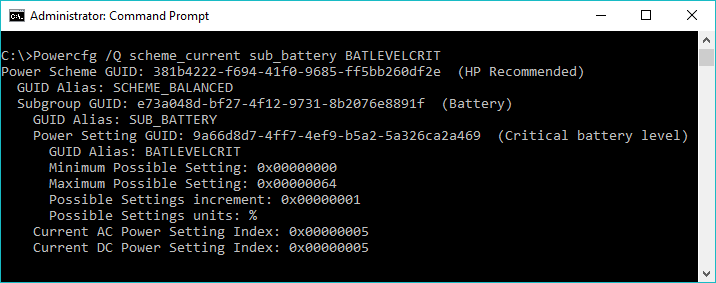
1. Launch an elevated command prompt and type the following commands to query the current values for the Low Battery Warning & Critical Battery values, respectively:  
   *Powercfg /Q scheme\_current sub\_battery BATLEVELLOW*

*Powercfg /Q scheme\_current sub\_battery BATLEVELCRIT*

Low Battery Level on DC Power is 9%:



Critical Battery Level on DC Power is 5%:

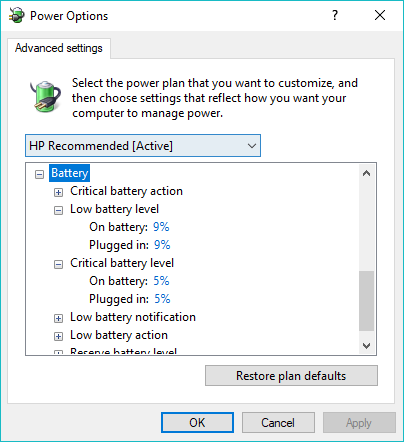


1. Take notes on what the current values are.

Note: these values must be converted from Hex to Decimal using the Programmer mode in the Calculator app.

## Step 2 – Confirm the Settings found by PowerCfg match the Control Panel UI

1. From the start menu, type “Power & Sleep Settings”
2. Click on Additional Power Settings
3. Click on Change Plan Settings
4. Click on Change Advanced Power Settings
5. Scroll down to and expand the Battery node
6. Expand the Low Battery Level and Critical Battery Level nodes.
7. Confirm the settings in the UI match the output from PowerCfg.



## Step 3 – Set New Critical Battery & Low Battery Values

1. Launch an elevated command prompt and type the following commands to set the Low Battery Warning & Critical Battery values, to 10% and 4%, respectively on both AC and DC:

powercfg.exe /setdcvalueindex scheme\_current sub\_battery BATLEVELLOW 10

powercfg.exe /setacvalueindex scheme\_current sub\_battery BATLEVELLOW 10  
powercfg.exe /setdcvalueindex scheme\_current sub\_battery BATLEVELCRIT 4

powercfg.exe /setacvalueindex scheme\_current sub\_battery BATLEVELCRIT 4

powercfg.exe /s scheme\_current

1. Repeat previous 2 steps to confirm the new settings were applied.

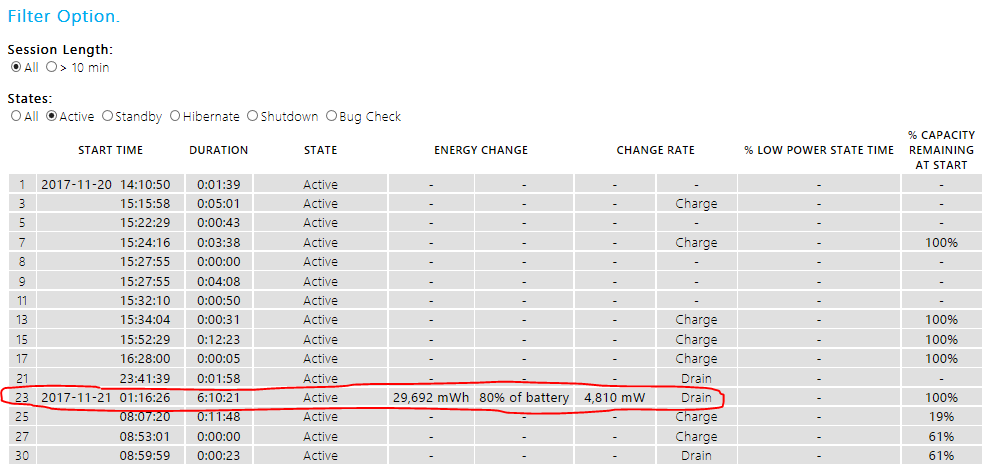
# Exercise 6B –PowerCfg System Power Report For Optimizing Screen On Power

**Estimated Time**: 5-10 min

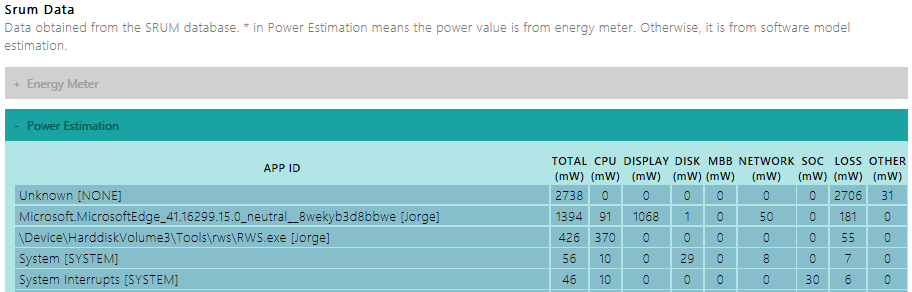
System power report (PowerCfg /SPR) provides power users and engineers with insights into the sleep states including screen on, standby, and hibernate. In this exercise, we will look at a SPR log to determine if there is anything that can be done to optimize for battery life. For guidance on how to use SPR logs to optimize Modern Standby sessions, please refer to the Modern Standby Optimization lab. If you’re not sure whether your device supports Modern Standby or legacy S3, run the following command from an elevated command prompt: PowerCfg /A

## Step 1 – Load SPR log and Filter to Active Sessions

1. You can create your own SPR log by launching an elevated command prompt and running “*Powercfg /SPR”,* however, in this exercise we will load a SPR log that was included with the lab material: c:\BatteryLife\PowerCfg\sleepstudy-report.html
2. Scroll down to the States filter option and click on the Active radio button.
3. Click on Session #23



## Step 2 – Review the E3 SRUM Data

1. Notice the different brightness levels the user set throughout the 6 hour, 10 min session.
2. Scroll down to the Srum Data section and expand the Power Estimation node.  
   
3. Notice how the largest amount of energy was attributed to the Unknown bucket. This device does not have HW power meters installed and E3 was not able to attribute this energy to an app, so it ended up in the unknown bucket.
4. The user was streaming video from Youtube in the Edge browser. Notice how the Edge app appears in the 2nd row, with the majority of the energy attributed to the Display.
5. Also notice an unexpected background app named RWS.exe was consuming the 3rd most energy throughout this active user session. This is a real-world stress that was used for demo purposes to consume CPU energy in the background.

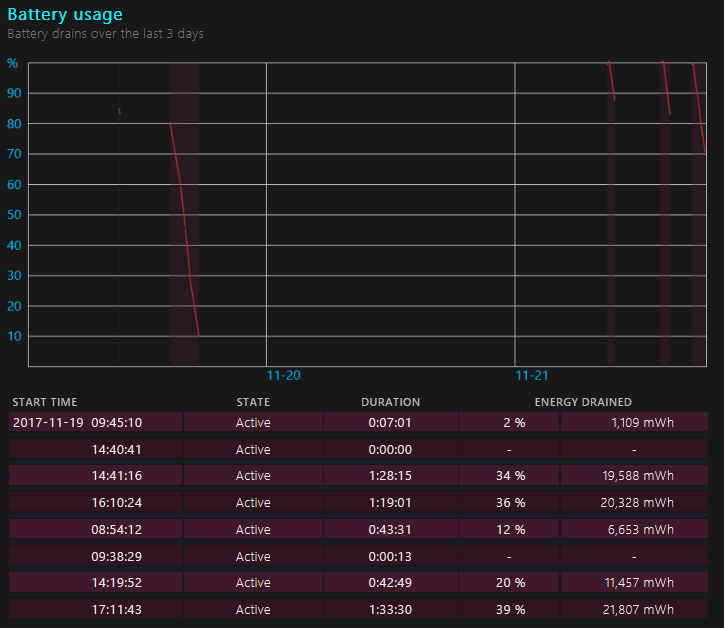
# Exercise 6C – PowerCfg Battery Report to Get Battery History & Stats

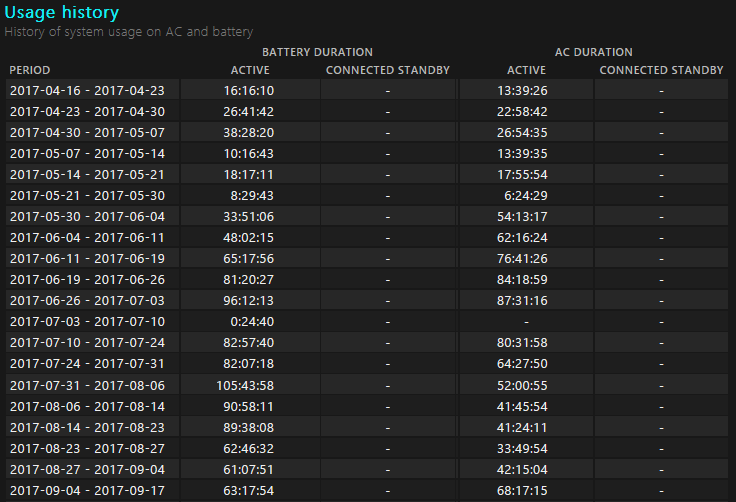
**Estimated Tim**e: 5 min

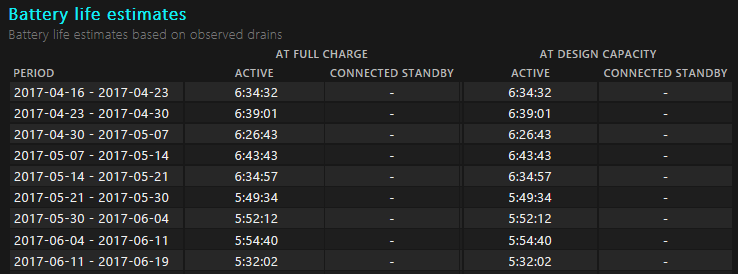
System power report (PowerCfg /BatteryReport) provides power users and engineers with insights the health of the battery and historical battery drain data.

## Step 1 – Generate and Review Battery Report

1. You can create your own SPR log by launching an elevated command prompt and running “*Powercfg /BatteryReport”,* however, in this exercise we will load a SPR log that was included with the lab material: c:\BatteryLife\PowerCfg\battery-report.html
2. Notice the following information offered in this report: Recent usage, Battery usage, Time spent on AC vs. DC, battery capacity history, and battery life estimates based on past drain rates.







# Exercise 7 – Identify Apps Reducing the System Timer Resolution in WPA

**Estimated Time**: 5 min  
**Level**: 300

Apps requesting low system timer resolutions (STR) such as 0.5ms or 1ms can impact battery life by more than 100mW. Additional context can be found in the Technical Background section of this lab.

## Step 1 – Generate WPR Log Using WPR.exe

1. You can create your own ETW log by launching an elevated command prompt and running the following commands. A sample ETW log is included in the supporting lab material.

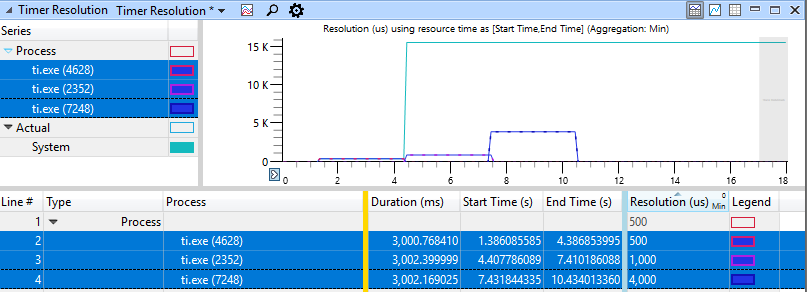
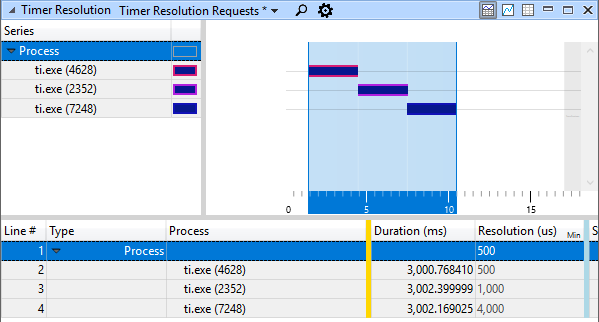
*wpr -cancel*

*wpr -start power -filemode*

1. Wait 60 seconds then run the following command:

*wpr -stop STR-IDLE.etl*

## Step 2 – Visualize the ETW log in WPA’s STR Views

1. From the start menu, type “WPA”, and select **Windows Performance Analyzer**
2. Load the trace captured in the previous step or the ETW log included in the lab material: c:\BatteryLife\WPA\STR.etl
3. Expand the power node
4. Scroll down and expand the Timer Resolution node
5. Double click on both the Timer Resolution and Timer Resolution Requests views.
6. In the Timer Resolution view, notice how the Y-axis in the graph is Timer Resolution plotted over time in the X-axis.
7. In the STR.etl sample provided, a test app called TI.exe is responsible for lowering the timer resolution to 0.5ms, 1ms, and 4ms in 3 second intervals.  
   
8. Scroll down to the Timer Resolution Requests view which shows the requests for lower timer resolutions by each process.  
   

Notes:

* If software components that are part of the OEM image are lowering the STR for prolonged periods, there may be an opportunity to improve battery life.
* The high resolution timers (HR Timers) listed in these views are internal to the Windows kernel. These will be separated into a separate view in a future drop of WPA.

# Annex I - Technical background

## Exploring Battery Saver & Battery Use By App UI

The Battery Saver user experience in Windows provides end-users with more control of battery life. It extends battery life by limiting background activity and reducing screen brightness by 30% once the remaining battery life reaches 20%. Learn more: <https://aka.ms/windowsbatterysaver>

The “Battery usage by app" user experience is user-configurable, allowing users to determine which UWP apps and services are allowed to run in the background. “Battery use by app” provides a stack ranked list of applications that impact battery life. This data is based on the energy attributed to applications by the Energy Estimation Engine (E3) service.

## Performance Power Slider User Interface

The Performance power slider enables users to choose the right balance between battery-life longevity and system performance. This user experience is easily accessible from the top-level UI by clicking on the battery system tray icon in the taskbar. The four slider positions on DC power are Battery Saver, Better Battery (formerly labeled Recommended), Better Performance, and Best Performance. When the user knows they won’t have access to AC power and wants longer battery life, they can move the slider to the Battery Saver or Better Battery position. When the user wants to tradeoff battery life for better performance, they can move the slider to Best Performance position. There are two main levers that are used behind the scenes to favor battery life or performance:

* Lever 1: Processor Power Management (PPM) tuning
  + Inbox PPM settings on all AMD and Intel chipsets, with or without Speed Shift.
  + PPM settings are OEM/ODM customizable via Windows provisioning.
  + Windows also supports processor specific PPM settings such as Intel’s Energy Performance Preference (EPP). Note: EPP is only supported on Intel devices with Speed Shift.
  + OEM firmware can be tuned for each slider position.
* Lever 2: Power throttling background applications
  + Reduces impact background apps have on CPU power by scheduling threads of background apps on cores that have a lower frequency.
  + Power throttling is not OEM configurable, however users can opt out apps via UI.
  + Enabled on all 6th+ gen Intel chipsets, with or without Speed Shift.
  + Enabling on AMD Ryzen mobile chipset is work in progress.
  + App developers can define power throttling levels using the SetProcessInformation function: <https://aka.ms/SetProcInfo>.

Learn more:

* Power Settings: <https://aka.ms/Config-Power-Settings>
* Power Slider: <https://aka.ms/Perf-Power-Slider>
* PPM Options: <https://aka.ms/PPM-Options>
* PerfEPP: <https://aka.ms/PerfEPP>
* Power Throttling Blog: <https://aka.ms/powerthrottling>

## Accessing & visualizing the Energy Estimation Engine data using PowerCfg

Energy Estimation Engine (E3) is a service running on all Windows devices that have a battery. It estimates and attributes energy consumption to apps. System Resource Usage Monitor (SRUM) is a framework that tracks per application resource usage in different time granularities. It maintains a local database to store the per-minute usage information up to two hours, and the per-hour aggregated usage information up to 60 days. SRUM providers in Windows are registered using SRUM infrastructure that collects, merges, and stores usage information about certain resources. “POWERCFG /SRUMUTIL” can be used to dump the energy estimation engine (E3) data stored in the SRUM database. Excel can be used to visualize the energy data and generate pivot tables that list out the top processes impacting power and energy consumption per hardware component. A variation of this exercise can be viewed in this online training video: <https://aka.ms/DefragTools-E3>

## Collecting and Visualizing E3 ETW events in WPA

Event Tracing for Windows (ETW) enables developers to instrument drivers and applications with performance events, start and stop event tracing sessions, and consume trace events. Trace events contain an event header and provider-defined data that describe the performance and energy characteristics of an application or operation. You can use the events to debug an application and conduct performance analysis. ETW tracing is very lightweight with minimal overhead per event, and the events are available in retail binaries.

This lab exercise will walk through the process of collecting an ETW log with the Microsoft-Windows-Energy-Estimation-Engine provider enabled and visualizing the events in the Windows Performance Analyzer (WPA) tool, which is part of the Windows Performance Toolkit (WPT).

## Battery Life Assessments

ADK Energy Efficiency workloads can be used to assess user experiences such as local full screen video playback (FSVP), streaming FSVP, and Office productivity. The workloads produce high level metrics such as “Projected time to shutdown”, E3 logs, and verbose ETW logs that can be analyzed in ETW consumer tools such as Windows Performance Analyzer (WPA) and Media eXperience Analyzer (MXA). These assessments have several pre-requisites including Office 2016, a Microsoft Account (MSA) for streaming in the Movies & TV app, as well as a Netflix subscription for the Netflix workload. These exercises can take several hours depending on how you configure the parameters to the assessments. For example, the assessment could start at 98% and rundown the battery to 5% (more accurate results) or you can run the assessment for shorter duration (less accurate results). See exercise below for details.

## Optimizing battery life for Low Power Media Playback Using MXA

There are several features in Windows that save power during full screen video playback. These features include audio offload, Multi-plane Overlay (MPO), Low Refresh Rate playback (48hz), and Independent Flip (iFlip). For example, when playing 24fps in full screen in Media Engine based app, the refresh rate drops from 60 to 48hz. This lab utilizes the ETW consumer tool called Media eXperience Analyzer to verify whether these power saving features are utilized by a Windows device. Learn more:

* <https://aka.ms/MXA-AudioOffload>
* <https://aka.ms/MXA-VideoPower>

## PowerCfg – Inbox Utility

PowerCfg is an inbox utility in Windows used by power users for setting, querying, and diagnosing power related settings and issues. Common usages include:

* Query and modify power settings using powercfg commands.
* System power report (PowerCfg /SPR) provides power users and engineers with insights into the sleep states including screen on, standby, and hibernate.
* Battery report (PowerCfg /BatteryReport) provides View battery drain history and battery condition info.
* E3 log (PowerCfg /SRUMUTIL) generates an Energy Estimation Engine (E3) log
* Energy report (PowerCfg /energy) collects a 60 second power trace and generates an energy report
* List available sleep states (PowerCfg /A) Prints out supported sleep states

Additional info:

* Overview: <https://aka.ms/DefragTools-PowerCfg>
* System Power Report 1: <https://aka.ms/DefragTools-PowerCfgSPR1>
* System Power Report 2: <https://aka.ms/DefragTools-PowerCfgSPR2>

## Capturing & Analyzing ETW logs with WPR and WPA

Windows Performance Recorder (WPR) enables power users and engineers to collect power traces that can be visualized in analysis tools. Windows Performance Analyzer (WPA) in the Windows Performance Toolkit (WPT) provides views that enable engineers to identify opportunities to optimize for power for all scenarios including system idle. The exercise in this lab introduces a new view that was added to identify applications that request low System Timer Resolutions (STR). The STR is expected to be at 15.6ms+ when on battery and system is idle. Time critical or real-time applications may request lower STR to reduce latency which impacts battery life. Low STR values can increase power consumption by > 100mW. Utilize WPR to collect ETW logs and WPA to verify software installed on OEM images does not unexpectedly lower the STR for prolonged periods.

# Annex II – E3 Logs field descriptions

|  |  |
| --- | --- |
| SRUMUTIL field | Definition |
| AppId | For modern applications, this gives the package name. For classic process, this gives the full path of image name. For service host, this gives svchost.exe [service category]. If the AppId with “EMI\_” prefix, this is not an application. Instead, it is a power rail that provides energy measurement through EMI interface. Learn more about EMI here: <https://aka.ms/EnergyMeteringInterface>  Note: when you see “unknown” appear in the AppId field, this is a bucket of energy that E3 could not account for when attributing energy to apps. |
| UserId | String name of user SID that runs this application. |
| TimeStamp | The time stamp of when this record saved to the database. Users can use the difference of the timestamp from the previous timestamp to find out the duration on which the energy is accumulated. On average, it should be one minute within the last 1-2 hours, and then one hour after that. |
| OnBattery | TRUE if the system is on DC at the recorded time. |
| ScreenOn | TRUE if the record is for screen on activity. |
| BatterySaverActive | TRUE if the record is for the period that Battery Saver is active. |
| LowPowerEpochActive | TRUE if the record is in modern standby and has passed the LowPowerEpoch phase. |
| Foreground | TRUE if the record is for foreground activity. |
| InteractivityState | The interactive state of the application during this record period.  Minimized: the application window is minimized  Visible: the application window is visible, but not focused  Focus: the application windows has the focus  NotUnique: at least two of the states listed above happened during the period |
| Committed | TRUE if the record has adjusted with all the known information. FALSE if the record is still subjected to change. |
| TimeInMSec | The record period in milli-seconds. |
| MeasuredBitmap | Bitmap to indicate if the component power is measured (1) or estimated (0). The 9 bits from left to right after “b” correspond to the 9 columns after MeasuredBitmap. |
| EnergyLoss | Energy transmission efficiency loss estimation in milli-joules. Energy loss is an artifact of power tree. When energy flows from source (battery) to the various rails, voltage is converted one or more times. Every time this conversion takes place energy loss occurs. Energy loss = .15 \* (CPU+SoC+Disk+Network+MBB). |
| CPUEnergyConsumption | Energy (milli-joules) consumed by the CPU. This is approximately 89% accurate on recent Intel chipsets. |
| SocEnergyConsumption | Energy (milli-joules) consumed by GPU. GPU energy is available only on Intel devices. GPU energy is not attributed to apps as of today. System wide GPU energy consumption is logged in system interrupt’s record. |
| DisplayEnergyConsumption | Energy (milli-joules) consumed by the display. |
| DiskEnergyConsumption | Energy (milli-joules) consumed by the disk. |
| NetworkEnergyConsumption | Energy (milli-joules) consumed by the network. |
| MBBEnergyConsumption | Energy (milli-joules) consumed by the mobile broadband. |
| OtherEnergyConsumption | Another name of Other Energy is Standby Activation Energy (milli-joules). Other Energy applies to Low Power Epoch (LPE) state only. It is estimated as the difference between (1) actual energy consumption by the device and (2) the least possible energy consumption by the device if it would otherwise be in Deepest Runtime Idle Platform State (DRIPS) for 100% of time. Other Energy is attributed to those apps that have one or more activators (that keep the device out of DRIPS) and their level of activity in LPE. |
| EmiEnergyConsumption | Energy (milli-joules) consumed by specific power rails that are being monitored by a hardware power meter. |
| CPUEnergyConsumptionWorkOnBehalf | Internal E3 field that does not contribute to total energy. A fraction of CPU energy consumed by this app (e.g., a Windows service) that relates to work executed by this app but is done on behalf of a beneficiary app (e.g., a UWP app). An app can work for itself (to realize its own responsibilities and/or use cases) or it can work on behalf of other apps (i.e., beneficiaries). It is possible for an app to work on behalf of many beneficiary apps. CPU Energy attribution algorithms appropriately take it into account to compute effective CPU Energy consumption of this app. It should not be subtracted again to compute total energy consumption of this app. |
| CPUEnergyConsumptionAttributed | Internal E3 field that does not contribute to total energy. Sum of CPU Energy consumed by other apps (e.g., Windows services) when they worked on behalf of this app (e.g., UWP app). An app can be benefited by work of many other apps. Those worker apps can attribute CPU Energy to the beneficiary app. CPU Energy attribution algorithms appropriately take it into account to compute effective CPU Energy consumption of this app. It should not be added again to compute total energy consumption of this app. |
| TotalEnergyConsumption | Total Energy is the sum of eight energy components: CPU Energy, SOC Energy, Display Energy, Disk Energy, Network Energy, MBB Energy, Other Energy, and Energy Loss. |