Hands-on Lab

Customizing & Optimizing

Modern Standby Devices

***Abstract*:** After your Modern Standby system is developed following the proper power management guidance, there comes the time of testing and validating that the power floor is optimized to deliver great battery life in standby. You might need to break down the system to isolate a power offender.

This lab will delve into useful tools such as System Power Report and WPA, and will guide you through various case studies illustrating commonly encountered problems. Topics include:

* Impact of USB devices
* Firmware problems and missing constraints
* Driver issues
* Identifying spurious wake sources
* New Adaptive Hibernate Triggers

Copyright (c) 2016 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 3](#_Toc499736422)

[Prerequisites 3](#_Toc499736423)

[Technical background 3](#_Toc499736424)

[Terminology 3](#_Toc499736425)

[Tools 4](#_Toc499736426)

[Exercise 1 – Navigating through a System Power Report 4](#_Toc499736427)

[Step 1: Obtain a System Power Report 4](#_Toc499736428)

[Step 2: Understand the System Power Report content 5](#_Toc499736429)

[Step 3: Look at System Information 7](#_Toc499736430)

[Step 4: Look at usage trends 7](#_Toc499736431)

[Step 5: Look at a session’s summary information 9](#_Toc499736432)

[Step 6: Look at session details 11](#_Toc499736433)

[Exercise 2 – Looking at Standby ETL traces with WPA 13](#_Toc499736434)

[Step 1: Gather a Standby trace with WPR 13](#_Toc499736435)

[Step 2: Viewing a Standby trace with WPA 14](#_Toc499736436)

[Step 3: Look at the Platform Idle State graph 14](#_Toc499736437)

[Step 4: Look at the DRIPS graph 16](#_Toc499736438)

[Identifying the Key Modern Standby Problems 18](#_Toc499736439)

[Exercise 3 – Identify problems with wakes 19](#_Toc499736440)

[Exercise 4 – Identify problems with missing drivers 20](#_Toc499736441)

[Exercise 5 – Identify problems with missing constraints 21](#_Toc499736442)

[Exercise 6 – Identify problems with USB devices 23](#_Toc499736443)

[Step 1: Identify with System Power Report 23](#_Toc499736444)

[Step 2: Identify with an ETL trace 24](#_Toc499736445)

[Appendix 1 – Adaptive Hibernate Triggers 26](#_Toc499736446)

[Adaptive Hibernate Trigger #1 – Standby Budget 27](#_Toc499736447)

[Adaptive Hibernate Trigger #2 – Active Screen on Reserve Time 27](#_Toc499736448)

[Recommended Doze to Hibernate timer value 28](#_Toc499736449)

[OEM Customization - Windows provisioning PPKG Settings sample 28](#_Toc499736450)

# Lab Goals

This lab will allow you to accomplish the following goals:

1. Interpret data from System Power Reports and WPA DRIPS plugins.
2. Identify common issues that can impact the power floor.
3. Understand the benefits of the adaptive hibernate triggers added to the latest version of Windows.

# Prerequisites

The Assessment and Deployment Kit (ADK) is available with OS releases.

Make sure you have the pre-generated results and test tools on your computer (located in **C:\Performance\StandbyLab**)

* SampleWorkload.etl
* USBProblem.etl
* Sleepstudy-report.html (4 different reports)

# Technical background

## Terminology

**Sleep = umbrella name for standby and hibernate**

* S3 = Legacy Standby power model
* MS = Modern Standby (S0iX) = umbrella name for CS & DS
  + DS = Disconnected Standby
    - Limited network connectivity \*
    - Continue listening to music and casting after entering standby
    - BT wake source support
    - Enables fast wake sources that are not available on S3 devices
  + CS = Connected Standby = AOAC (Always On Always Connected)
    - All features offered on DS
    - System is connected to network and network triggers device to wake
    - Enables end users to stay connected to email, social networks, VOIP, and receive push notifications in standby
* S4 = Hibernate = Deeper sleep state than standby (S3 or MS)  
    
  \* Exception made to enable uninterrupted audio playback and Miracast scenarios

Deepest runtime idle platform state (DRIPS) is when the system is consuming the lowest amount of power possible, limited by the system’s power floor. When the screen is turned off, the Modern Standby session starts and the system goes through multiple phases to move into a low-power state. When the system is in the lowest-power state, the system is in DRIPS. The system is not in DRIPS when performing tasks like receiving emails, updating live tiles with fresh content, receiving VoIP calls, or any other background task that requires system resources. The more time the system spends in DRIPS before the screen is turned back on, the longer the battery life.   
  
Activators   
Activators are software components that are allowed to perform work in the background while in Modern Standby.

Total standby session time = DRIPS time + non-DRIPS time

## Tools

**System Power Report** is a Windows diagnostics tool that supports Modern Standby (connected or disconnected). It monitors a Modern Standby PC’s behavior and provides actionable diagnostics on Modern Standby battery life. It’s available on Modern Standby capable PCs as well as S3 systems. System Power Report generates a summary of top issues causing poor Modern Standby battery life and give an overview of all power states a device experienced.

The **Windows® Performance Toolkit** consists of two independent tools: **Windows® Performance Recorder (WPR)** and **Windows® Performance Analyzer (WPA)**. WPR is a powerful recording tool that creates Event Tracing for Windows (ETW) recordings. You can run WPR from the user interface (UI) or from the command line (CL). WPR provides built-in profiles that you can use to select the events you want to record. WPA is a powerful analysis tool that combines a flexible UI with extensive graphing capabilities and data tables that can be pivoted and have full text search capabilities.

# Exercise 1 – Navigating through a System Power Report

**Estimated Time**: 10 minutes

Windows includes a built-in utility for tracking Modern Standby battery performance with minimal impact: **System Power Report**. **System Power Report** provides overview information about each time the system is in Modern Standby, including active time, idle time, and power consumed. **System Power Report** also provides first-level information of activity during each Modern Standby session, allowing for easy investigation of long-running activities. SPR will also include sessions such as Active, Hibernate, Bug checks, and other power states.

## Step 1: Obtain a System Power Report

System Power Report is simple to use and run from an Administrator Command Prompt (also referred to as an elevated command prompt). System Power Report outputs an easy-to-read HTML report. To run System Power Report, type any of the following commands into an Administrator Command Prompt:

powercfg.exe /SPR

powercfg.exe /SystemPowerReport

powercfg.exe /Sleepstudy

The built-in powercfg.exe utility will create an HTML file named *sleepstudy-report.html* in the current working directory.

**Note**: You can skip this step for the WinHEC lab, as the following steps use pre-generated reports.

## Step 2: Understand the System Power Report content

Open the pre-generated **C:\Performance\StandbyLab\sleepstudy-report.html** report with your favorite browser.

The System Power Report is organized around each system state as one *Session—* A Modern Standby session is one instance of the screen turning off and then back on again. Modern Standby data is only included in 10+ min sessions to ensure the system has had enough time to reach DRIPS, you can filter to greater than 10 min session with the Filter options near the top of the SPR. For each session, an overview is provided at the top with a hyperlink to a detailed section for each session later in the HTML report.

The System Power Report also contains:

* Static configuration information about the system, its OS installation and firmware version
* A graphical view of the usage trend over the last 72 hours
  + You can modify the time period using the **/duration** parameter
* A summary table of each system state session, including
  + Session start time, and duration
  + State description
  + Power source (AC or battery power)
  + Battery power consumed and average power consumption
  + Software and Hardware DRIPS % (MS sessions only)
  + Battery % at start of session (MS sessions only)
* Detailed information for each Modern Standby session, including
  + Information contained in the summary table
  + Reasons for Connected Standby entry and exit viewable from the drop down menu on the right, see table 1 for all entry and exit reasons
  + Histogram chart of idle time
  + Top five active components during the Modern Standby session, including the type, name, and device path (if applicable)
* Source Resource Utilization Monitor (SRUM) Data. 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.  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).
* Information on the system battery configuration, including design capacity and cycle count

|  |  |
| --- | --- |
| **Code** | **Reason** |
| 1 | Unknown |
| 2 | Power Button |
| 3 | Remote Connection |
| 4 | SC\_MONITORPOWER |
| 5 | User Input |
| 6 | AC/DC Display Burst |
| 7 | User Display Burst |
| 8 | PoSetSystemState |
| 9 | SetThreadExecutionState |
| 10 | Full Wake |
| 11 | Session Unlock |
| 12 | Screen Off Request |
| 13 | Video Idle Timeout |
| 14 | Policy Change |
| 15 | Sleep Button |
| 16 | Lid |
| 17 | Battery Count Change |
| 18 | Grace Period |
| 19 | Graphics PnP Operation |
| 20 | Dynamic Partitioning |
| 21 | Hibernate, or Shutdown |
| 22 | System Idle Timeout |
| 23 | Proximity Sensor |
| 24 | Thermal Standby |
| 25 | Resume PDC |
| 26 | Resume S4 Display Burst |
| 27 | Termina |
| 28 | PDC Signa |
| 29 | AC/DC Display Burst Suppressed |
| 30 | Transition to System Sleep State S4/S5 |
| 31 | Winrt API |
| 32 | Input Keyboard |
| 33 | Input Mouse |
| 34 | Input Touch |
| 35 | Input Pen |
| 36 | Input Accelerometer |
| 37 | Input Hid |
| 38 | Input UserPresent |
| 39 | Input SessionSwitch |
| 40 | Input Initialization |
| 41 | PDC Signal: Windows Mobile Power Notification |
| 42 | PDC Signal: Windows Mobile Shel |
| 43 | PDC Signal: Hey Cortana |
| 44 | PDC Signal: Holographic Shell |

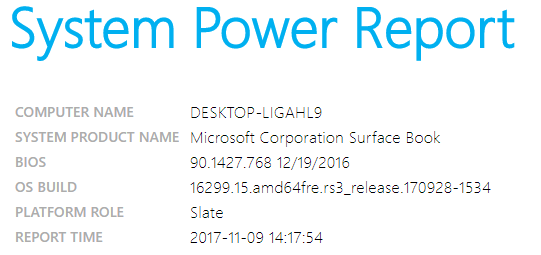
Table 1. enumerating reasons for Modern Standby entry and exit

The remainder of this overview walks through a sample System Power Report highlighting how to interpret the report and the information it contains.

## Step 3: Look at System Information

Every System Power Report begins with basic system information, including system name and firmware version. This information is critical to be included in each report because OS, firmware, and BIOS changes can have significant impact on Modern Standby battery life.

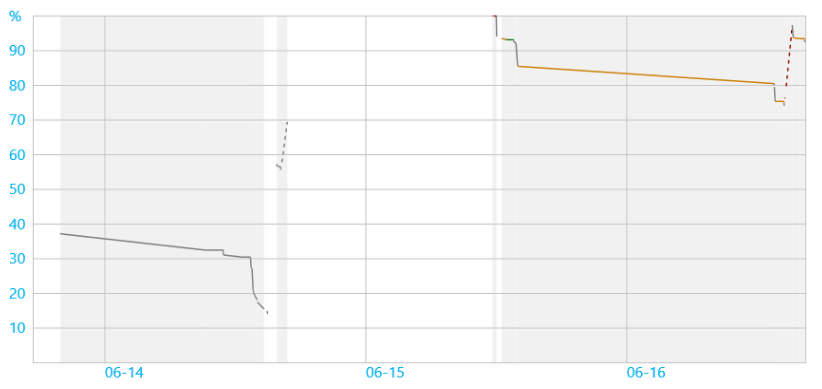
The screen shot below shows the system information included in each System Power Report:



This example shows that this is a Microsoft Corporation Surface Book and running Windows 10 build 16299.

## Step 4: Look at usage trends

Every System Power Report also includes a graphical view of system usage, including Modern Standby periods.

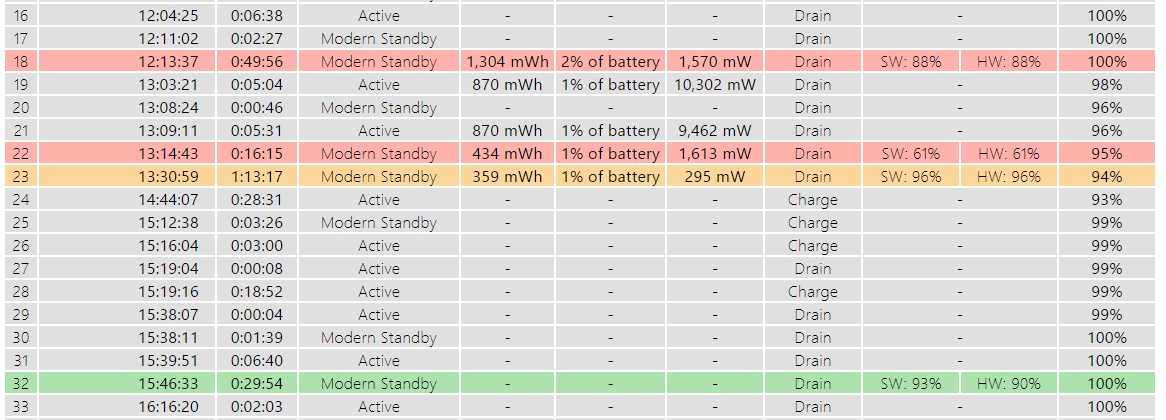


* The chart is color-coded with green, orange, and red segments corresponding to low, medium, and high system activity
* The chart includes dotted, solid, and no line formats corresponding to AC power, Battery Power, and System Off periods
* The default period covered by the chart is 3 days
  + You can modify the default time period using the **/duration** parameter. For example: powercfg /spr duration 4 (sets duration to 4 days)

The chart legend is included with each System Power Report.

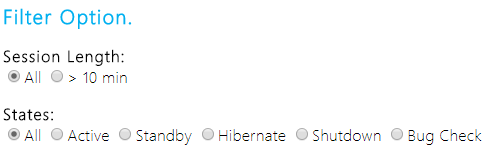
## Step 5: Look at a session’s summary information

Each System Power Report also includes a tabular summary of the Modern Standby sessions evaluated in the report.



Each row in the summary table contains information about one State session, this can be Active, Standby, Hibernate, Shutdown, or Bug Check. The Modern Standby rows in the summary table are color-coded to identify sessions that should be investigated for improvement.

Above this view are filer options that can be used to quickly limit the sessions shown in the table.



In the example above, you can see that the power floor of the system varies greatly between sessions, with a minimum of 359 mW, and a maximum of 1,304 mW. The high-power floors need to be investigated as they will negatively impact the system’s Standby battery life.

The color coding for each session is based on a combination of the *drain rate* and the *deepest runtime idle platform state (DRIPS)* rate (percent low power). The color is determined by the worst performance of either drain rate or DRIPS rate, according to the following rules:

* **Red** indicates at least one of the following:
  + The DRIPS rate is < 80 percent.
  + The drain rate is >= 1 percent per hour. (If the drain rate is >= 1 percent per hour, the Modern Standby battery life will be at most four days.)
* **Orange** indicates at least one of the following:
  + The DRIPS rate is between 80 and 94 percent.
  + The drain rate is between 0.333 and 1 percent. (If the drain rate is under 0.333 percent, the platform will achieve 12+ days battery life in Modern Standby.)
* **Green** indicates all other cases.

It is not unusual for some sessions to be red or orange because of activity, but you can expect most sessions to reflect low activity and thus low power consumption (green). The color scheme is designed to make potential high-drain sessions easy to identify. You should be most concerned about long sessions (of several hours) that show high sustained battery drain—these sessions have the most impact on the battery and are mostly likely to indicate an OS and/or hardware issue.

System Power Report calculates drain rate by using remaining capacity information that is provided by the platform battery and charging subsystem. The battery capacity, in milliwatt-hours, is recorded at the start and end of each System Power Report session. The battery capacity information is provided by the platform through the ACPI \_BST control method under the battery device object in the ACPI namespace.

The summary table includes the following basic information (from left to right):

|  |  |
| --- | --- |
| Column | Description |
| **Session Number** | Starts with one, and increments for each session that is reported during this period. The default report period covers the last three days. |
| **Start Time** | Local time when the screen turned off for this session. Shown in YYYY-MM-DD HH:MM:SS format. Additional sessions on the same day do not repeat the year-month-day information. |
| **Duration** | This duration is the approximate amount of time passed between the transition to *screen off* and subsequent transition to *screen on.* |
| **State** | The system state that this row reflects, Active, Standby, Hibernate, Shutdown, or Bug Check |
| **Energy Change** | Shows the number of absolute milliwatt-hours (**mWh**) that are consumed and the relative percentage of the battery's last full-charge capacity. A session in which no change occurs in remaining capacity or the battery was charging is indicated by a hyphen (**-**). |
| **Change Rate** | In milliwatts, and AC (**Charge**) or DC (**Drain**) power source indicator. The change rate is calculated by dividing the **Energy Change** value by the **Duration** value. |
| **% Low power state time** | Shows the software DRIPS rate and (if applicable) hardware DRIPS as the percentage of time in which the SoC resides in the lowest power state (DRIPS). Hardware DRIPS (denoted by **HW:** preceding the residency percentage) is available on Intel and Qualcomm SoC-based Windows PCs only. |
| **% Capacity Remaining at start** | Displays the battery % the device was at when it entered the state – this is avalible for Active and Modern Standby states. |

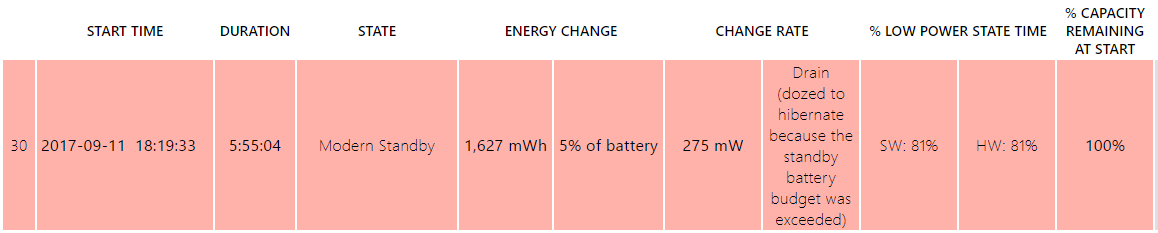
To measure Modern Standby performance, session duration should be greater than ten minutes. Longer periods (more than one hour) are reflective of real user experience and have the most significant impact to battery life. Energy change is not tracked for AC sessions.

All other states besides Modern Standby will be colored grey.

## Step 6: Look at session details

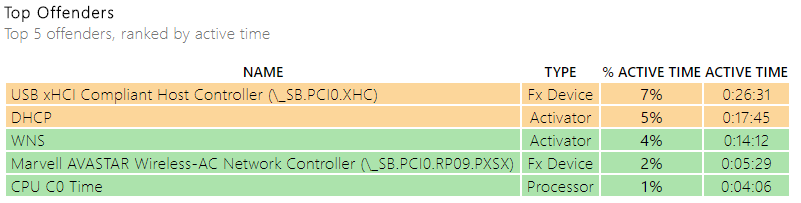
Open **C:\Performance\StandbyLab\sleepstudy-report(2).html**

Every row in the summary table is a hyperlink to per-session details that are presented in the System Power Report. Click on session 30, which shows a 1,627 mW power drain, and the lowest DRIPS percentage at 81%.



Below the summary information, the top five offenders are listed. Top offenders are the components that are responsible for the most active time during the Modern Standby session.

The table of top offenders displays the component name, type, relative active time, and absolute active time for each top offender, as shown in the following example.



In this example, a USB xHCI Compiant Host Controller is active for about 45 minutes. We will consider this problems like this in detail in exercise 4.

Problems are caused by issues with the devices, drivers, or bad power management implementation. Additional investigation can then focus on those devices using tools from your silicon vendor, an instrumented platform, a kernel debugger, etc.

The table can show the following types of offenders:

* **Fx Device.** A device that has a driver that implements support for the Windows power framework (PoFx).
  + <https://msdn.microsoft.com/en-us/library/windows/hardware/hh406637(v=vs.85).aspx>
* **Activator.** A software component that can keep the system active to perform valuable work during Modern Standby.
* **PEP Pre-Vetoes**. Power Engine Plugin gives information about the Soc subsystem that veto's Modern Standby and prevents the system from entering DRIPS, please contact your silicon vendor for further information
* **SoC Subsystems**. System of Chip subsystems (e.g. GPU, DSP), to which the OS has no visibility that are responsible for power drain while the system is in DRIPS
* **Processor.** CPU active time that occurs outside of an activator being enabled.
* **PDC Phase.** Time that is spent in the different phases of entering or exiting Modern Standby.

Each component in the **Top Offenders** table is color-coded by activity time. If it is overactive for more than ten percent of the session, the row is highlighted in red. If the component is active between five and ten percent, it is colored orange. Otherwise, the component row is highlighted in green.

# Exercise 2 – Looking at Standby ETL traces with WPA

**Estimated Time**: 15 minutes

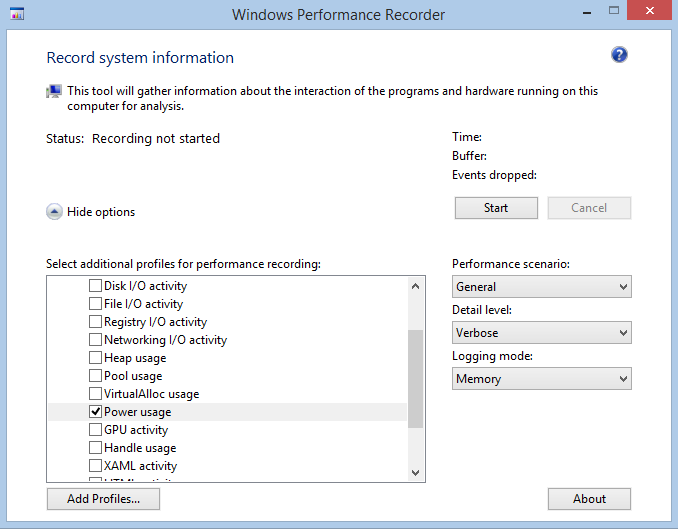
The Windows Performance Analyzer (WPA) allows you to view traces of system activity in a graphical format. WPA is used for many Windows performance and debugging scenarios and is the second-level triage tool for Modern Standby issues which cannot be resolved using System Power Report. WPA presents a graphical format of a trace file which contains events collected during a Modern Standby session.

## Step 1: Gather a Standby trace with WPR

Capturing trace is the key diagnostic method used to debug issues observed during Modern Standby through System Power Report or other tools. A trace contains detailed information on system platform states, device states, software activity, CPU utilization, memory utilization, and other system events. Using the events captured in a trace, you can observe exactly what happened during Modern Standby and the resulting problems.

It is recommended that a trace be captured for at least one hour of Modern Standby to allow enough time to observe trends and averages.

1. Open **Windows Performance Recorder (WPR)** from the **Start** menu.
2. Select the **Power usage** profile.



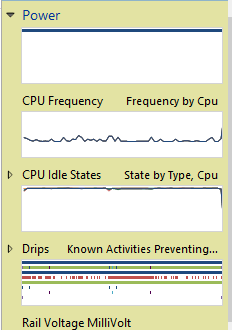
1. Press **Start** to begin recording the trace.
2. While recording, put the system into **Modern Standby**.
3. Wait for at least one hour and then wake up the system.
4. Press **Save** to stop recording.
5. Choose a file name on the dialog that appears.
6. Enter at least one letter into the description field and press **Save**.
7. Wait for the progress bar to complete and then press OK.
8. If **WPR** is still Recording, press **Cancel**.
9. Retrieve your saved trace file for analysis.

**Note**: You can skip this step for the WinHEC lab, as the following steps use pre-generated reports.  
You can find the traces on the lab computer in: C:\Performance\StandbyLab

## Step 2: Viewing a Standby trace with WPA

You can use the WPA tool to view and analyze Modern Standby traces.

1. Launch **WPA** from the **Start** menu.
2. From the **File** menu, click **Open** and select a trace file.
   1. Choose **C:\Performance\StandbyLab\SampleWorkload.etl**
3. You can add graphs to the current analysis view from Graph Explorer by following these steps:
   1. Expand the **Power** category in **Graph Explorer**.
   2. Select the graph you want to add and drag it to the analysis view pane.

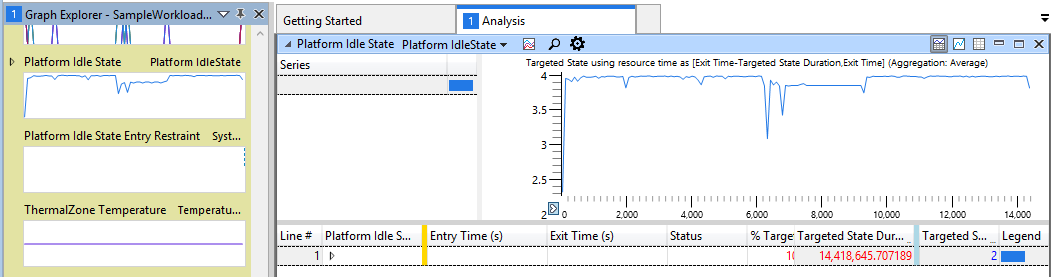


## Step 3: Look at the Platform Idle State graph

Note: If you have a problem dragging the Platform Idle State graph to the analysis window, this is a known bug, please right click on Platform Idle State and select "Add all platform Idle State graphs to new Analysis Window"

This graph shows the platform wide idle state plotted against time. On different platforms, the numerical states may correspond to different states; platform-specific documentation should provide a mapping. You should contact your silicon partner to find out which platform idle state is DRIPS for your SoC.

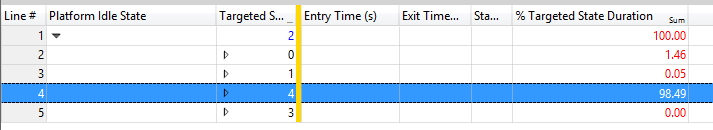
The SoC in the example below has State 4 as DRIPS and State 0 as active. The higher numbers mean lower power consumption states, with the highest number representing DRIPS.



You can use this graph to figure out the DRIPS % which is the percentage of time when the system is in DRIPS during the session. This is important data to know because it has a direct effect on battery life.

If the DRIPS % is high, battery life is longer than if DRIPS % was lower in most cases because it means that the system spent more time in a low power state.

1. Drag the **Targeted State** column next to the **Platform Idle State** column.
2. Expand the **Platform Idle State** row. This will tell you the % of time the system was in each state
3. Look at the % Targeted State Duration value for Targeted State == 4.



In this example, the DRIPS % is 98.49%, which is considered excellent.

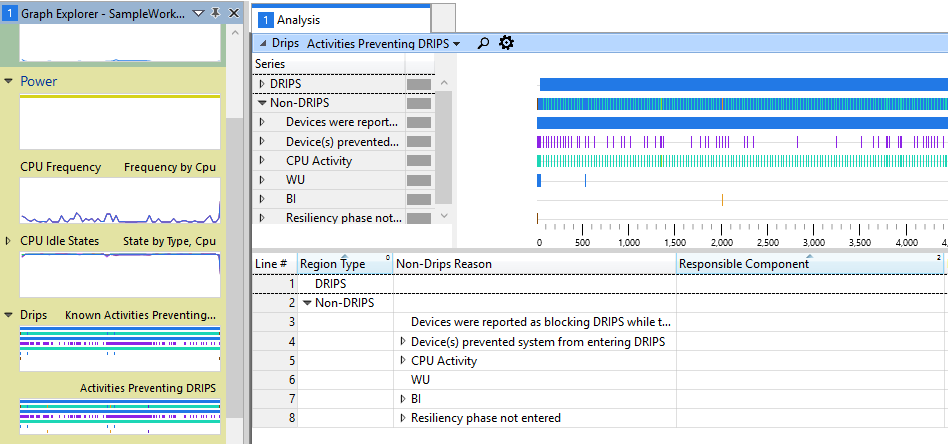
DRIPS % evaluation guide

| DRIPS % | Evaluation |
| --- | --- |
| 98 – 100 | Excellent |
| 95 – 97.9 | Very good |
| 90 – 94.9 | Good |
| 80 – 89.9 | Fair |
| < 80 | Poor |

There are some exceptional cases where even DRIPS % is high and the battery life is poor. For example, a DRIPS % of 95 or higher can still result in less than three days of battery life. Those cases are likely due to hardware problems and should be investigated further to understand the root cause. They can also be easily identified in System Power Report as sessions with high software and hardware DRIPS rates, but have a high drain rate and are colored red.

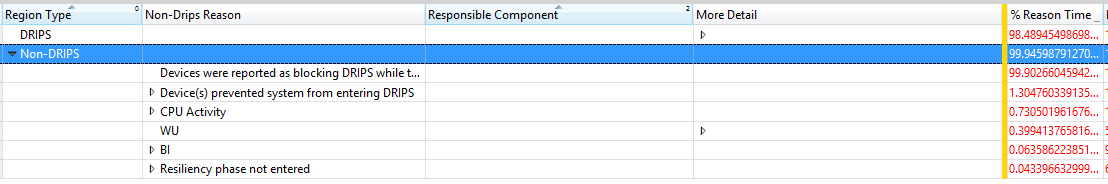
## Step 4: Look at the DRIPS graph

The *Activities Preventing DRIPS* graph shows the components that are active during the trace period including activators, devices, and processes. You can use this graph to figure out the components that are active the longest which prevented the system from entering DRIPS.



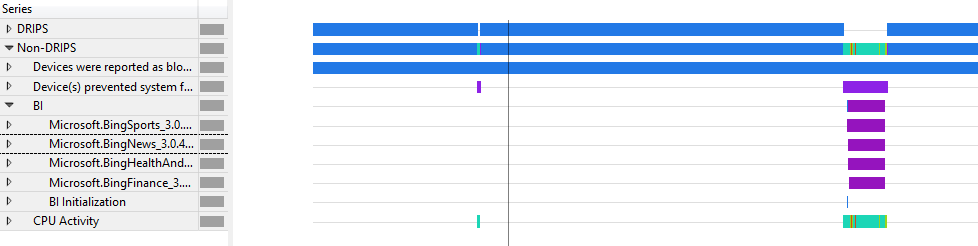
Activators are components that could take references and perform tasks while in Modern Standby. Some common ones are BI (Background infrastructure), WNS (Windows Notification Service), and WU (Windows Update). You can use this graph to figure out the top active Activator during the session. This is important because a particular activator could be holding a reference for long periods of time which prevented the system from entering DRIPS.

Components that are shown in this graph, except for Devices and CPU Activity, are activators. The example above shows BI and WU as activators. You can figure out the top activators by opening the table view and looking at the *% Reason Time* column which shows the percentage of time the activator was active during the session.

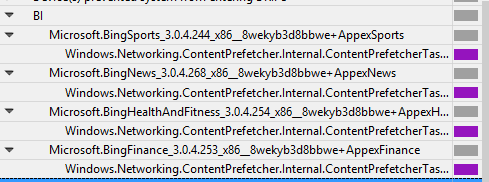


BI is a special activator because it provides broker services to apps to access system resources. When BI shows up as an active activator, you can expand the BI row and figure out which apps are causing BI to be active. You can use this graph to figure out the top active apps during the session.

The example below shows an example of four Bing apps that are causing BI to be active.



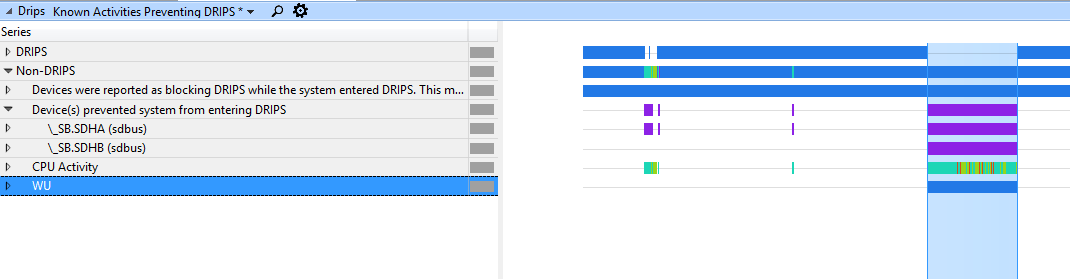
To drill deeper into the apps, you can expand each app in the graph view and it will show you information on the individual tasks within the app that are keeping the app alive. In this example, the apps are all simply prefetching content from the network to keep the apps fresh.



Besides activators, another possible reason that the system could not enter DRIPS is because some devices are actively running. The name of the devices is hardware-dependent. You can obtain the device names for your platform by looking at the ACPI DSDT table. Similar to system idle states, devices have states. Devices have different states ranging from D0 to D3. Device states are standardized across platforms.

You can use this graph to figure out the top active devices during the session. This is important because a particular device could be holding a reference for long periods of time which prevented the system from entering DRIPS. Keep in mind that some devices may be active because an Activator is running some tasks that requires them to be active; for example. Background update downloads will cause the network device to be active.

In this example, Windows Update is keeping the SD bus active (on which is connected the storage).



## Identifying the Key Modern Standby Problems

There are several problems which may prevent a system from consistently entering DRIPS. A system cannot enter DRIPS if there are any tasks running that require the system to stay active. Similarly, the system cannot enter DRIPS if there are any SoC devices or connected devices powered on. The analysis steps below will walk you through the process to figure out which tasks or devices are causing non-DRIPS time.

1. The first thing to look at is resiliency activations. Look at the DRIPS graph and figure out the top active Activators. If there are none, skip to step 4.
2. Locate the top active activator and get the % of active time by looking at the graph table.
3. If the % of active time is significant, you should notify your Microsoft contact to understand why the activator is preventing the system from going into DRIPS. If the top Activator is BI, you should expand the BI section in the System Power Report or the ETL trace to determine the active application. You can then supply this information to Microsoft directly and figure out why those apps are preventing the system from going into DRIPS.
4. If the % of active time by Activators isn’t significant or not present, the next thing to look for is devices. Look at the DRIPS graph and figure out the top active devices and the % of time they’re active.
5. If the % of active time is significant, it’s indicative that device is the reason why the system is not going into DRIPS. If you’re a hardware manufacturer, you should contact your device vendors to understand why they are preventing the system from going into DRIPS.
6. If neither Activators nor devices present a significant problem, there’s probably an issue with the underlying hardware. You should contact your silicon partner or leverage a power instrumented platform to determine where the power drain comes from.

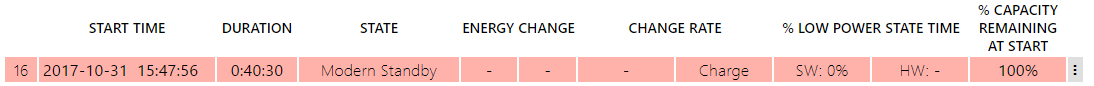
# Exercise 3 – Identify problems with wakes

**Estimated Time**: 5 minutes

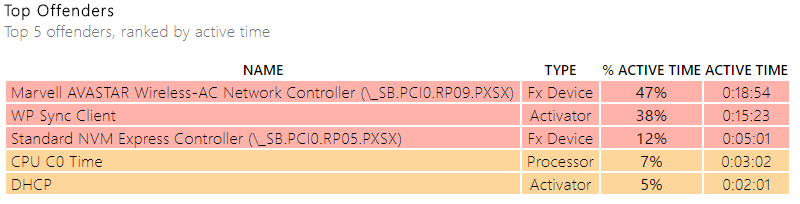
Devices should avoid unexpectedly waking up the SoC through interrupts (for example, interrupt storms, bad debouncing, etc.). This causes the system to exit DRIPS, increases the average power floor and reduces battery life. This is known as spurious wakes.

The analysis process for networking devices is straightforward.

1. Open the pre-generated **sleepstudy-report(3).html** report with your favorite browser.
2. Click on Session 16.



1. Look at the **Top Offenders** table. The networking wireless device is listed as active 47% of the time during the session.



1. Click on the Fx Devices row to obtain details about this offender.



Surprise wakes and Power Transitions are clearly identified by the report. There are 2,354 of them in this example. A follow-up discussion should happen with the IHV to determine why the wireless adapter is waking up the SoC unexpectedly. The underlying problem could be a bad device firmware implementation.

# Exercise 4 – Identify problems with missing drivers

**Estimated Time**: 5 minutes

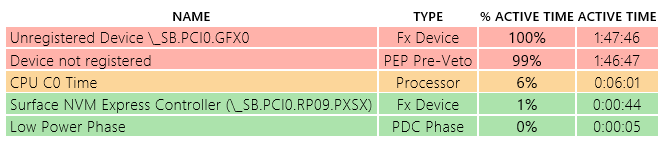
A tight integration between drivers, devices, Windows, and the firmware is required to ensure proper power management and a high rate of DRIPS residency.

If a device is missing a driver or doesn’t have the right driver, it can impact the associated device power management and resulting D-state.

1. Open the pre-generated **sleepstudy-report(3).html** report with your favorite browser.
2. Click on Session 53. The system consumes 12.995 Watts of energy during in 1 hour 47 minutes and the DRIPS % is 0.



1. Look at the **Top Offenders** table. An **Unregistered Device** is listed as active 100% of the time during the session.



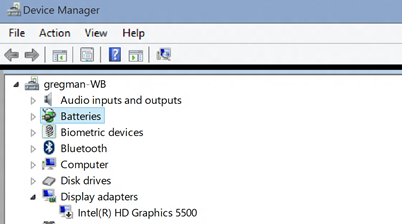
The unregistered device is the graphics adapter (GFX) as specified by the device name \_SB.PCI0.GFX0.

When an **Unregistered Device** is present in the top offenders list, it means that the ACPI firmware has defined it, but Windows doesn’t have it on its list of power managed devices.

This could mean one of two things:

* One device is missing a driver.
* An ACPI firmware constraint was defined for a device not present in the system.

In this example, the system didn’t have a proper driver installed for the **Display Adapter**, as shown below in the **Device Manager**.



# Exercise 5 – Identify problems with missing constraints

**Estimated Time**: 5 minutes

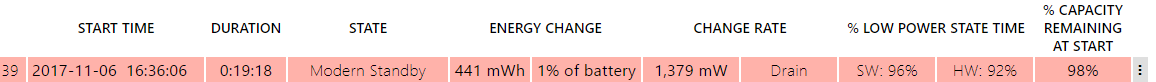
The SoC power state is the sum of the states of all the devices.

Windows keeps a list of devices and their states that are critical to reach low power – they are called constraints. Windows will wait for all constraints being met before engaging resiliency and enter DRIPS. The constraints are specified by an OEM and the SoC vendor through the ACPI firmware.

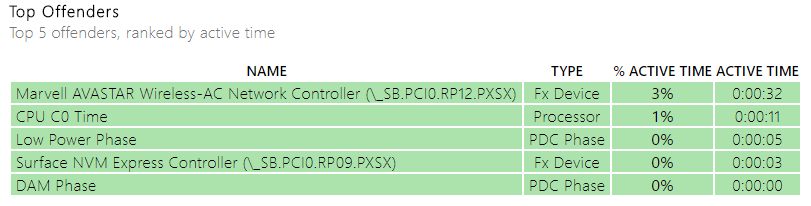
The ACPI firmware must be modified if an OEM changes the SoC vendor reference design and constraints must reflect those changes accurately.

Missing constraints, or too many constraints, can cause a variety of problems that increase power drain during standby.

1. Open the pre-generated **sleepstudy-report.html** report with your favorite browser.
2. Click on Session 39.
   * The system has a high change rate of 1379 mW
   * The software DRIPS % is 96%
   * The hardware DRIPS % is 92%

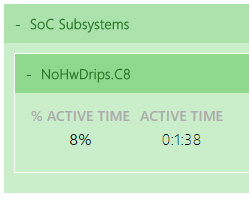


1. Look at the **Top Offenders** table
   * There are no top offenders for this session, however we have lost a lot of battery



Large discrepancies between DRIPS % (for example 96% SW and 92% HW) and high battery loss is usually symptomatic of a missing constraint in the ACPI firmware.

Simply stated, Windows assumes the system is ready to enter DRIPS, but some hardware component is still active and prevents the SoC package from entering S0 low power idle. The SoC Subsystems section can give clues to what hardware components might be active and what state the SoC was in during the session.



The next logical step is to try to isolate and identify which hardware component is still in D0 and consuming power by using either a power instrumented platform or your silicon partner debugging tools.

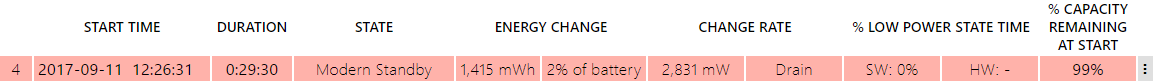
# Exercise 6 – Identify problems with USB devices

**Estimated Time**: 10 minutes

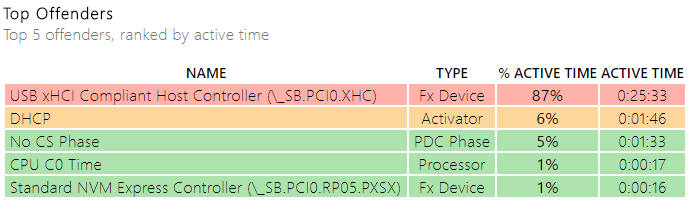
The USB host controllers can power down only after all of the devices connected to them have entered a low power state. This means that USB devices must support selective suspend on Modern Standby devices to ensure the SoC can entire DRIPS while the screen is off.

## Step 1: Identify with System Power Report

1. Open the pre-generated **sleepstudy-report(2).html** report with your favorite browser.
2. Click on Session 4. The system consumes 2.83 Watts of energy during 29.5 minutes and the DRIPS % is 0.



1. Look at the **Top Offenders** table.
   1. The USB host controller (\_SB.PCI0.XHC) is active for 87% of the session duration.
   2. XHC is the USB3.0 host controller.



When the USB bus controller is active for minutes at a time in Modern Standby, it usually means that one USB device attached to the bus is not entering, or doesn’t support selective suspend. The next logical step is to determine which USB device is staying in D0 by looking at an ETL trace.

In this SPR you can continue opening the children of the USB host controller to find the child at fault, in this case it’s a USB Audio Device.



MSDN documentation on selective suspend:  
<https://msdn.microsoft.com/en-us/library/windows/hardware/ff540144(v=vs.85).aspx>

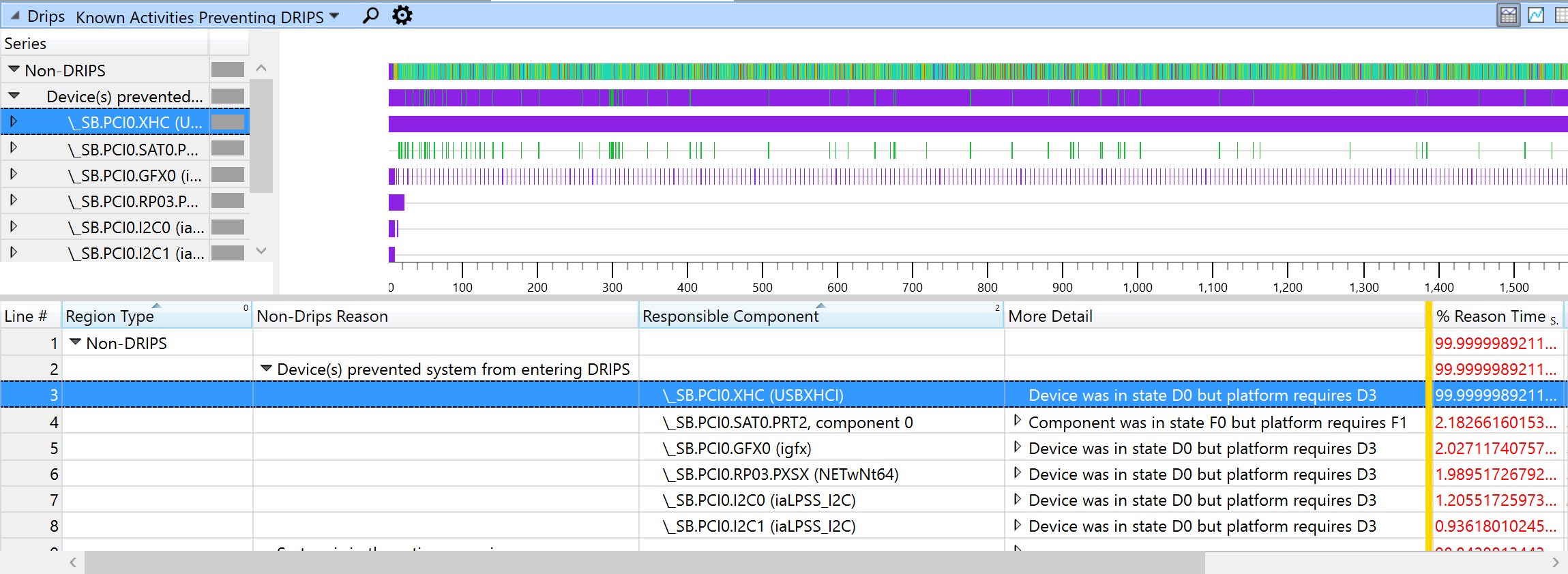
## Step 2: Identify with an ETL trace

Another option to furthering a USB investigation is with an ETL trace. You would need to capture one on the same system where the System Power Report was generated. This trace was captured by WPR, as described earlier in the document.

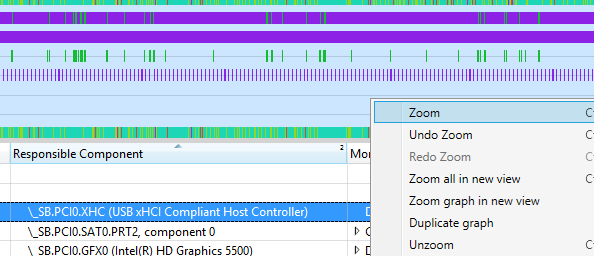
**Note**: This trace is a USB issue but different then the example shown in **sleepstudy-report(2).html**

To investigate USB issues, you’ll use the **DState** graph and table.

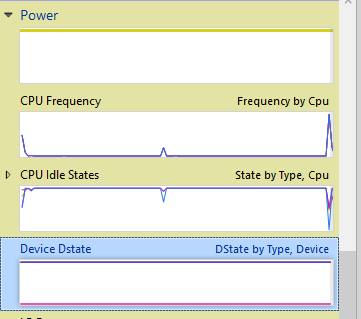
1. Open **USBProblem.etl** with WPA.
2. Drag and drop the **DRIPS** graph in the analysis tab.
3. Look at the **Non-Drips Reasons**, and find the USB xHCI Host controller as a device preventing the system from entering DRIPS.
   1. You can see that the device is active for 99% of the trace (as shown in the **% Reason time** column).



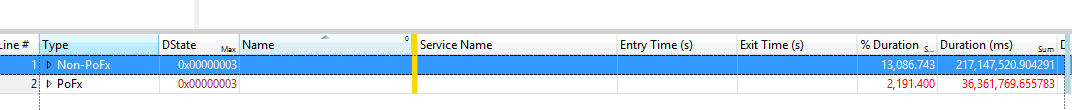
1. Zoom in the region where the USB xHCI Host controller is active.
   1. Select the device in the table.
   2. Right-click on the light blue interval in the graph, and select zoom.
   3. The **% Reason time** should now be 100%.



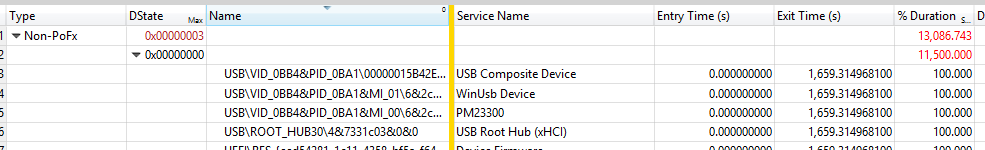
1. Find the **Device Dstate** graph under the **Power category** of the **Graph Explorer**.



1. Drag and drop the **Device Dstate** graph in the analysis tab.
   1. The Device DState graph shows devices effective D-States over time. You can use the data to determine if a specific device enters the appropriate D-state while the system is in Modern Standby.
      1. **PoFx** type is for devices managed by the Windows Power Management Framework.
      2. **Non-PoFx** type. This is where the USB-attached devices will be found.
2. Move the **DState** column right next to the **Type** column. You viewport should look like this:



1. Expand the **Non-PoFX** category.
2. Expand the **Dstate** row with the 0x0 value (D0 state, or active).
3. Sort by the **Name** column and find the USB devices.



The data in the D-state table shows that, while the system was in standby, a USB composite device was still in state D0 for 100% of the time. The hardware id of the composite device is USB\VID\_0BB4&PID\_0BA1\00000015B42EE80F0000000000000000. This is the device that was preventing the XHCI controller from powering down.

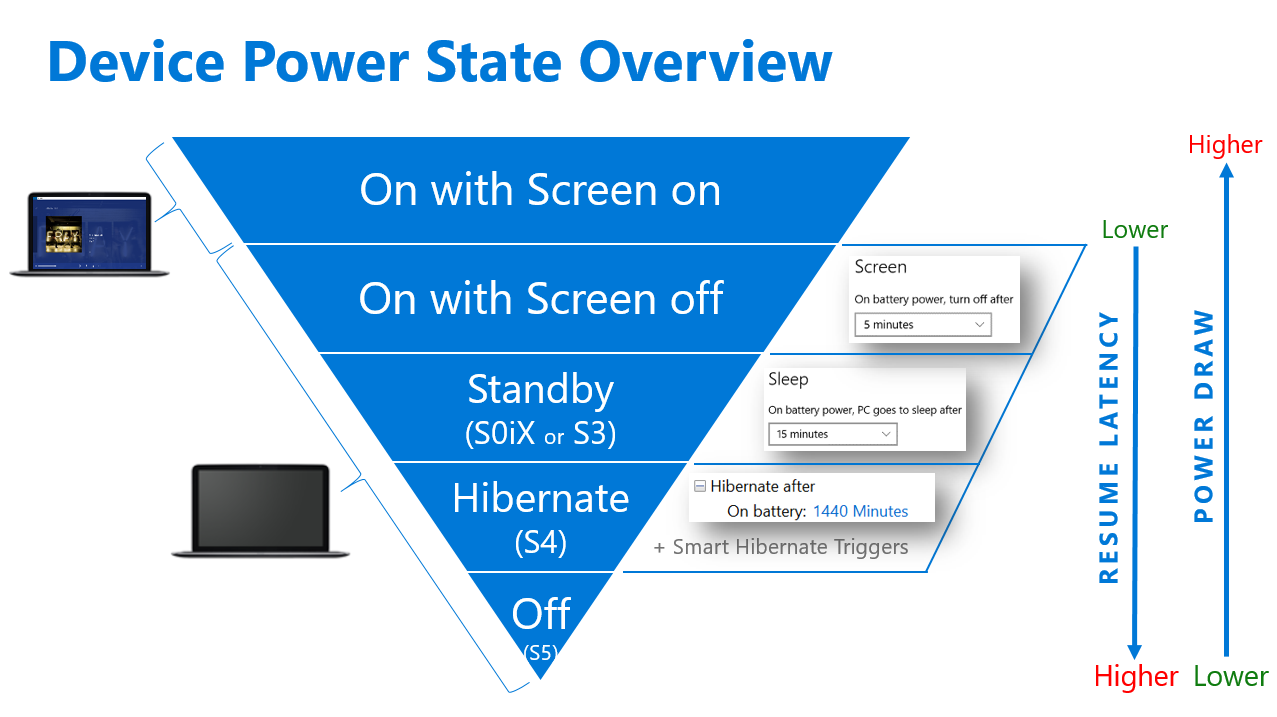
If the device is managed by a driver authored by Microsoft, please report the issue to Microsoft. If not, then this information must be reported to the hardware vendor who owns the driver to find a solution and ensure the device enters selective suspend.

# Appendix 1 – Adaptive Hibernate Triggers

Hibernate is an option that is leveraged by users to put the system into a low power state when the system is not in use. The current logic for hibernate relies on an OEM or user configured doze to hibernate timer. The most common timer value observed has been 4 hours in the Windows 10 timeframe. A fixed doze to hibernate timer does not address rapid drain of battery nor delivers the best user experience on devices that have a very low standby power floor because the device may prematurely enter hibernate and prevent the user from taking advantage of the benefits that Modern Standby offers.

In Windows 10 Anniversary Update edition, new triggers were added to allow the system to hibernate intelligently. These new hibernate triggers are like airbags in a car, the goal is that you do not have to use them but they can be triggered in cases where the power floor in modern standby is high. The new triggers and their objectives are:

* **Standby Budget Trigger**: Provide a great modern standby experience by ensuring that the system remains in modern standby for as long as possible when the power floor is low.
* **Active Screen-on Reserve Time Trigger**: Eliminate the chances of resuming to a dead battery.

Before going into the details of the new adaptive hibernate triggers, please review the following screenshot which provides a visual in how hibernate relates to other device power states:  


A

## Adaptive Hibernate Trigger #1 – Standby Budget

The following settings are used to set the standby budget; this is the amount of battery the device can drain in modern standby before transitioning to hibernate.

|  |  |  |
| --- | --- | --- |
| Budget Settings | Definition | *Default values* |
| STANDBYBUDGETPERCENT | Defines the battery drain % that the user is allowed in standby session | DC: 5 %  AC: 0% (disabled) |

## Adaptive Hibernate Trigger #2 – Active Screen on Reserve Time

Active screen-on reserve time is the amount of battery time the user will be guaranteed to experience after resuming from hibernate. In other words, the user should never return to their device and find a dead battery because the device remained in modern standby to the point where the battery completely drained. The following settings are used to set the reserve time:

|  |  |  |
| --- | --- | --- |
| Budget Settings | Definition | *Default values* |
| STANDBYRESERVETIME | Defines the minimum screen on time that will be available to the user after standby exits and screen turns on | DC: 20 mins  AC: 0 mins (disabled) |

Based on our experiences with windows insider community the recommended value is 20 minutes.

## Recommended Doze to Hibernate timer value

Microsoft recommends that doze timeout to greater or equal to 24 hours (1440 minutes).

## OEM Customization - Windows provisioning PPKG Settings sample

The above defined triggers are configurable via windows provisioning framework. To configure the power settings using the [Windows provisioning](https://msdn.microsoft.com/en-us/library/windows/hardware/dn916154(v=vs.85).aspx) framework, you must create a [Windows provisioning answer file](https://msdn.microsoft.com/en-us/library/windows/hardware/dn916153(v=vs.85).aspx) as one of the inputs to the Windows Imaging and Configuration Designer (ICD) command-line to generate either a provisioning package or a Windows image that contains the power settings. For information on how to use the Windows ICD CLI, see [Use the Windows ICD command-line interface](https://msdn.microsoft.com/en-us/library/windows/hardware/dn916115(v=vs.85).aspx).

The following example XML shows what your Windows provisioning answer file might look like after you've written it to configure adaptive hibernate settings:

<?xml version="1.0" encoding="utf-8"?>

<WindowsCustomizatons>

<PackageConfig xmlns="urn:schemas-Microsoft-com:Windows-ICD-Package-Config.v1.0">

<ID>{XXXX GUID}</ID> <!-- ID needs to be be unique GUID for the package -->

<Name>CustomOEM.Power.Settings.Control</Name>

<Version>1.0</Version>

<OwnerType>OEM</OwnerType>

</PackageConfig>

<Settings xmlns="urn:schemas-microsoft-com:windows-provisioning">

<Customizations>

<Common>

<Power>

<Policy>

<Settings>

<AdaptivePowerBehavior>

<SchemePersonality>

<Default SchemeAlias="Balanced">

<Setting>

<!—Set Adaptive Hibernate settings -->

<StandbyBudgetPercent>

<DcValue>25</DcValue> <!—20 Perrcent -->

</StandbyBudgetPercent >

<StandbyReserveTime>

<DcValue>300</DcValue> <!-- 20 minutes -->

</StandbyReserveTime>

</Setting>

</Default>

</SchemePersonality>

</AdaptivePowerBehavior>

</Settings>

</Policy>

</Power>

</Common>

</Customizations>

</Settings>

</WindowsCustomizatons>

Additional information on adaptive hibernate can be found on MSDSN: <https://msdn.microsoft.com/en-us/library/windows/hardware/mt732711%28v=vs.85%29.aspx>