Lab5: Power in WDM

# Overview

In this lab we'll implement power capabilities using WDM

## Setup

### Start this lab from lab4. Copy lab4 to lab5, and rename lab4.sln to lab5.sln

### On the target machine enable hibernation using powercfg.exe utility

### On VmWare enable sleep: <http://www.vmware.com/support/ws55/doc/ws_special_acpi_guest.html>

# Power Dispatch Routine – IRP\_MN\_SET\_POWER

## SwdmDispatchPower

### Open power.c. There is a skeleton of SwdmDispatchPower.

### Using IoGetCurrentIrpStackLocation, get a point to the current stack, and call it pStk

### Alias pointer to device context (device extension) to pCtx.

### Get the MinorFunction from the stack location into a variable. Name it IRP\_MN\_Code

### Acquire IoAcquireRemoveLock the remove lock on the device extension. Release the lock at the end.

### Switch on IRP\_MN\_Code

### Create cases for IRP\_MN\_SET\_POWER, IRP\_MN\_QUERY\_POWER, IRP\_MN\_WAIT\_WAKE and IRP\_MN\_POWER\_SEQUENCE

### Build and test. Put a breakpoint on SwdmDispatchPower and step through.

## The IRP\_MN\_SET\_POWER

### Log (using DbgPrintEx) information about the Irp.

### Based on the stack location – there are two cases. pStk->Parameters.Power.Type may be SystemPowerState or DevicePowerState. Fork the code.

if (pStk->Parameters.Power.Type == SystemPowerState)

### Within the device context there is a copy of the power state (pCtx->SysPwrState). Copy the system power state from the Parameters (Power.State.SystemState)

### Copy current stack location to next one. IoCopyCurrentIrpStackLocationToNext

### Set completion routine to CompletionSetSystemPower (to be defined later)

### Mark IRP as pending.

### Call the next driver, using IoCallDriver.

### Return STATUS\_PENDING

### For the case of DevicePowerState:

### See if we are powering up or down. Our previous state is stored in pCtx->DevPowerState. Compare it with Parameters.Power.State

### We need to distinguish between power up and power down. The following code will do:

if( (pCtx->DevPwrState >= pStk->Parameters.Power.State.DeviceState) ||

(pStk->Parameters.Power.State.DeviceState == PowerDeviceD0))

### Log using DbgPrintEx

### Mark the Irp as pending: IoMarkIrpPending

### Copy the current stack location to next: IoCopyCurrentIrpStackLocationToNext

### Set a completion routine to CompletionDevicePowerUp (To be defined later)

### Call the next driver, and return STATUS\_PENDING

### For power down: Identify that we are powering down with the following if:

if( ( pStk->Parameters.Power.State.DeviceState > pCtx->DevPwrState ) ||

( pStk->Parameters.Power.State.DeviceState == PowerDeviceD3 ) ||

( pStk->Parameters.Power.ShutdownType == PowerSystemSleeping3 ) ||

( pStk->Parameters.Power.ShutdownType == PowerActionHibernate ) ||

( pStk->Parameters.Power.ShutdownType == PowerActionShutdown ) ||

( pStk->Parameters.Power.ShutdownType == PowerActionShutdownReset ) ||

( pStk->Parameters.Power.ShutdownType == PowerActionShutdownOff ) )

### The new device power state is at: pStk->Parameters.Power.State.DeviceState. Copy device power state to context (DevPwrStatte).

### Tell the system that you are switching to a new power state. Call PoSetPowerState with the new power state.

### We now call the next device driver in the stack. We don't need to setup a completion routine. To do this, call IoSkipCurrentIrpStackLocation, and call the next device in stack, and then IoCallDriver.

### Release the remove lock.

## CompletionSetSystemPower

This is the completion routine, when we call the lower driver in respond to set system driver for the system.

### Check the status of the IRP: Irp->IoStatus.Status

### If not success, complete the request, with IoCompleteRequest.

### Otherwise, get stack location to pStk and context to pCtx

### Put the system state pStk->Parameters.Power.State.SystemState into a variable called Index. We'll use this index within the Global\_PowerInfo\_Ptr->DeviceState array.

### Get the desired device state from Global\_PowerInfo\_Ptr->DeviceState[index], into a local variable PowerState (this is a union, use the DeviceState).

### The context has a PowerIrp field. Save the Irp there.

### Call PoRequestPowerIrp to fire a power IRP for device. Use the following code:

Status = PoRequestPowerIrp(

DeviceObject,

IRP\_MN\_SET\_POWER,

PowerState,

(PREQUEST\_POWER\_COMPLETE)CallBackForSetPower,

Context,

NULL);

We'll define the callback routine in a minute.

### If status is pending, return STATUS\_MORE\_PROCESSING\_REQUIRED

### Otherwise, release the remove lock and complete the request.

## CallbackForSetPower

### Get pStk and pCtx

### Save the system power state and the device power state in the context:

index = pStk->Parameters.Power.State.SystemState;

pCtx->SysPwrState = pStk->Parameters.Power.State.SystemState;

pCtx->DevPwrState = Global\_PowerInfo\_Ptr->DeviceState[DevicePowerIndex];

### Release the lock and complete the request.

## CompletionDevicePowerUp

### Take stack location and context into pStk and pCtx

### Copy the device state to the context pCtx->DevPwrState

### Notify the system about the new state: PoSetPowerState

### Release the remove lock.

# Power Dispatch Routine – IRP\_MN\_QUERY\_POWER

## Overview

### For Query Power have a case IRP\_MN\_QUERY\_POWER

### Just like Set Power, fork between the two cases. When pStk->Parameters.Power.Type is SystemPowerState and when it is DevicePowerState

### Log using DbgPrintEx

## For SystemPowerState

The system is asking us whether it can switch the system power state. We do nothing but passing the request to next driver:

IoMarkIrpPending

IoCopyCurrentIrpStackLocationToNext

### IoSetCompletionRoutine to CompletionQuerySystemPower (to be defined below)

### And finally: IoCallDriver

## For DevicePowerState

For device power query, we also pass the request down. The only difference is a different completion routine. Use CompletionQueryDevicePower (to be defined below)

## CompletionQuerySystemPower

### When the lower driver returns from a system query power, we need to issue a device query power.

### Get current stack location and context: pStk, pCtx

### Get the desired device state from the global power info array (same as above)

### Call PoRequestPowerIrp:

Status = PoRequestPowerIrp( DeviceObject, IRP\_MN\_QUERY\_POWER, PowerState,

### If status is pending, return MORE\_PROCESSING\_REQUIRED. Otherwise, release the remove lock and complete the request.

## CallbackForRequestPower

This is a callback for the device setup we issued on the completion routine in 3.4. We simply release the remove lock and complete the Irp

## CompletionQueryDevicePower

### Release the remove lock and complete the request