USB-C™ and Power Delivery Architecture in Windows 10

Jr-Chiang (JC) Jaw
Microsoft Corporation, Senior Software Engineer

USB Developer Days 2017

Taipei, Taiwan

October 24-25, 2017

Agenda: USB-C and PD Architecture in Win10

- Introduction
  - When does the OS need to be involved in USB-C and PD?
- Windows 10 USB-C architecture
  - UCSI (USB Type-C™ Connector System Software Interface)
  - Ucm (USB Connector Manager)
  - UcmTcpci (USB Connector Manager TCPCI extension)
- USB-C troubleshooting notifications
- USB-C and PD Testing
  - HLK tests required for Win10 certification
  - USB-C and PD test tools
- Future USB-C/PD Support
- Q&A

Introduction

- When does the OS need to be involved?
- Roles and responsibilities of the OS vs. the hardware/firmware

The goal… “USB-C just works”

When the user connects two USB Type-C devices together, it will always work as intended

In order to achieve this goal, software, firmware, and hardware must all work together

When does the OS need to be involved?

- Scenarios that require UI feedback to the user
  - Example: Troubleshooting UI notifications
- Scenarios that require contextual information from the user or from the OS
  - Example: Configure USB-C/PD settings
  - Example: OS-specific power or data role preferences
- Systems without PD controllers that need software TCPMs
- Other general benefits:
  - Software can more easily and quickly be upgraded compared to hardware or firmware
    - Faster fixes
    - New features

Roles and responsibilities

- Windows 10:
  - Provide troubleshooting messages
  - Perform SKU-specific PD policies (e.g. role swap)
  - Enable advanced users to configure USB-C/PD settings
  - Provide software TCPM for systems without PD controllers

- Hardware/firmware:
  - Provide accurate state of hardware
  - Provide ability for Windows to control the hardware
  - Provide basic default policies that don't require contextual information provided by the OS
    - Example: As a sink, always negotiate and charge at the highest available rate
    - Example: DRP devices that are meant to be used in a certain role should always attempt to get into that role

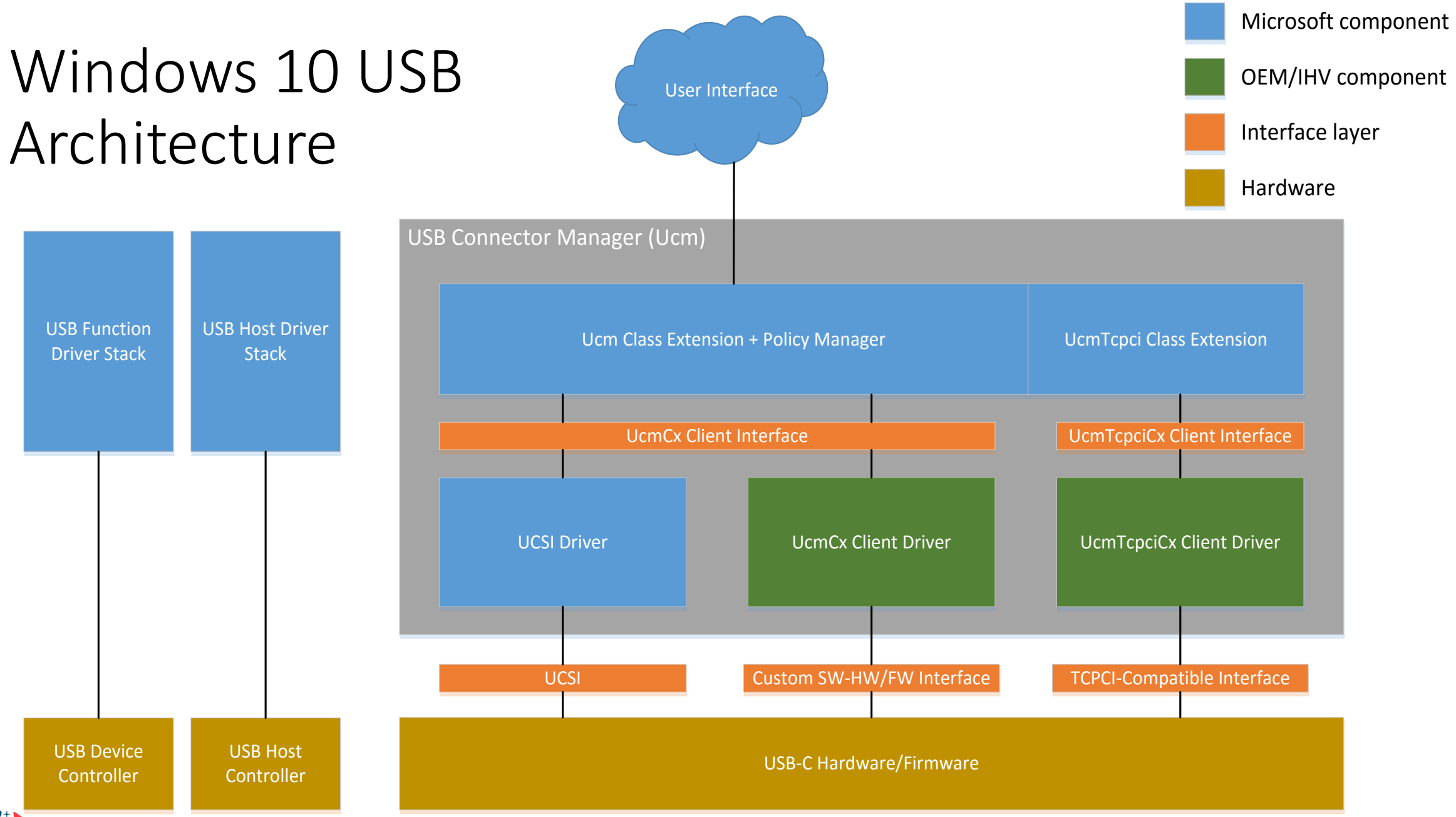
Windows 10 USB-C Architecture

- Overview of the USB Connector Manager
- UCSI
- Ucm Class Extension
- UcmTcpci Class Extension

USB Connector Manager

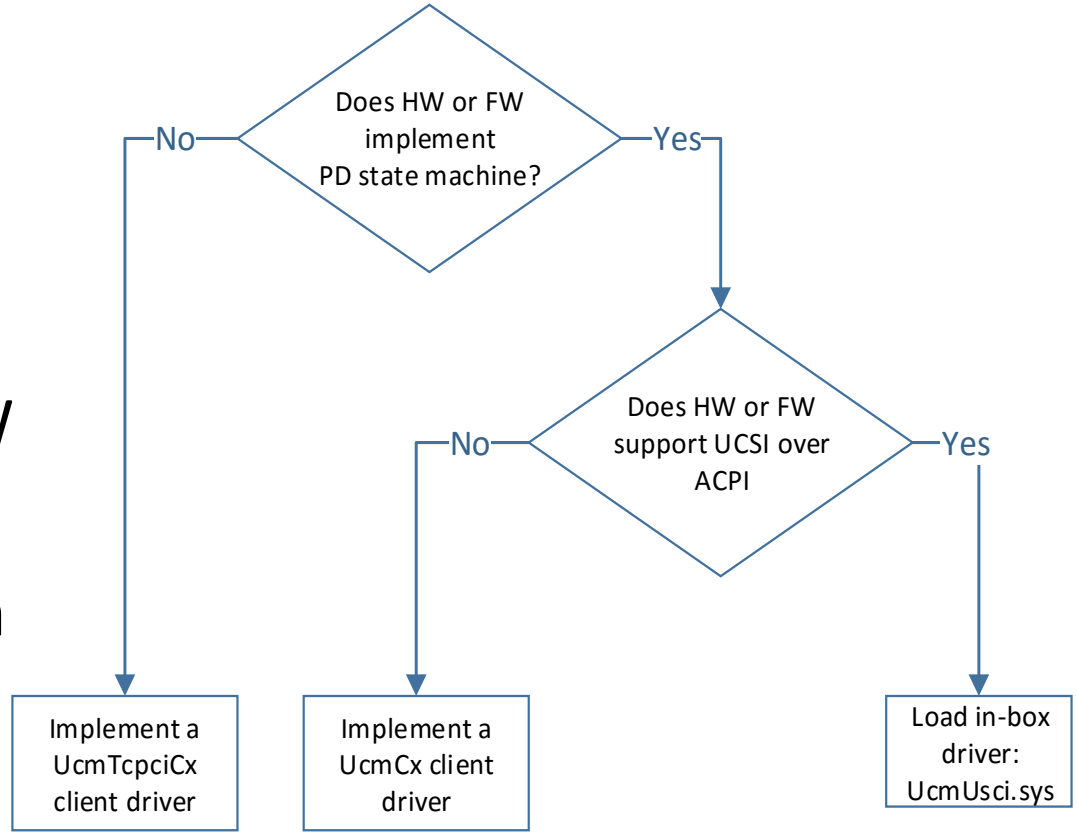
- What is USB Connector Manager (UCM)?
  - Name of the Windows 10 driver stack that manages USB-C/PD for the system
  - Its role is to get hardware status and makes policy decisions
  - It interfaces with other UI components in the OS (e.g. settings, notifications)
- Hardware/firmware has multiple options for plugging into UCM
  - Option 1: Support UCSI (USB Type-C Connector System Software Interface)
    - No driver development required (but EC-development is required)
  - Option 2: Write a UcmCx client driver
    - Use the UcmCx driver template on Windows Drivers Github
  - Option 3: Write a UcmTcpci client driver
    - Use the UcmTcpci driver template on Windows Drivers Github

Windows 10 USB Architecture



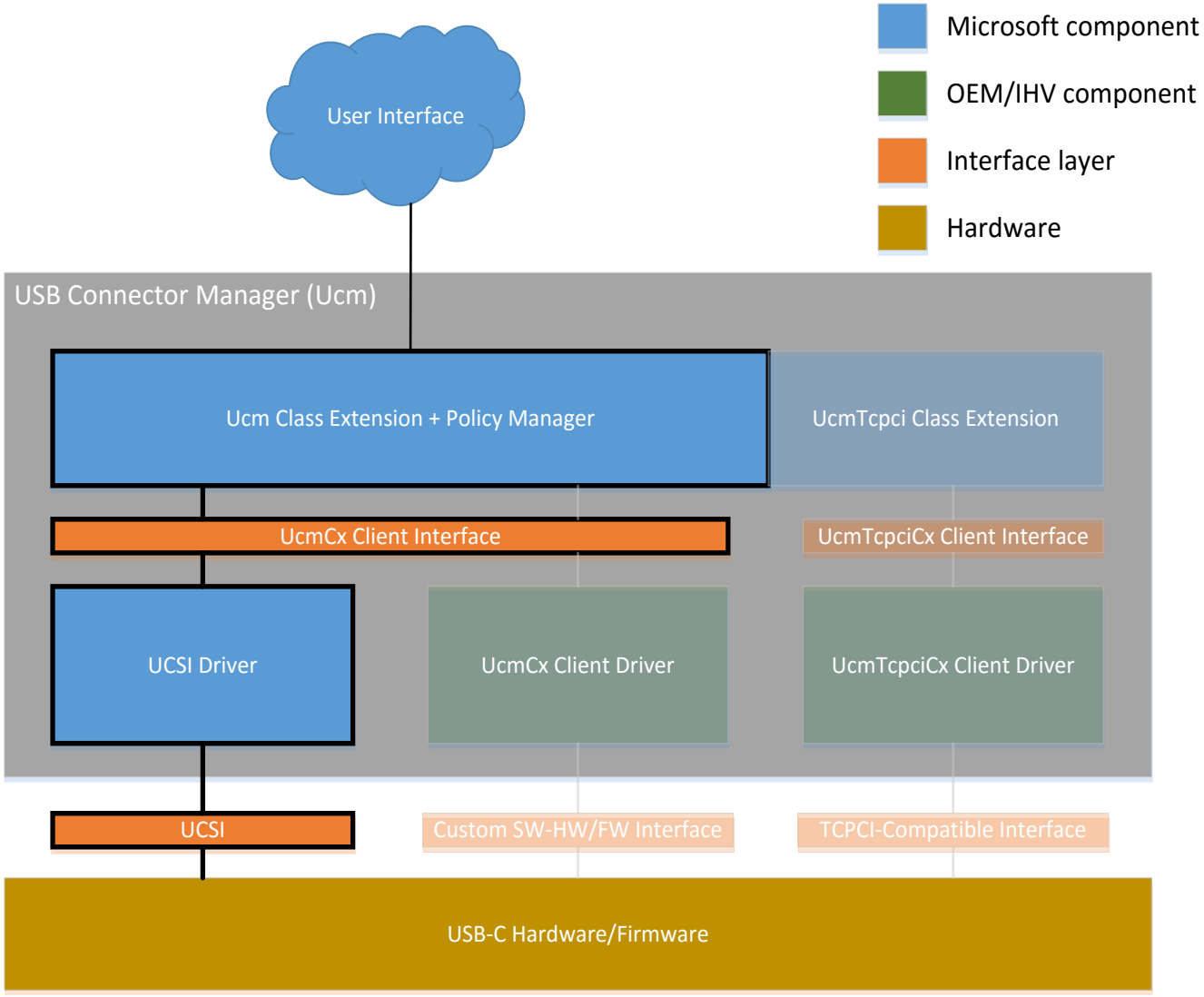
Choosing the Right Software Model For Your HW

- Use UcmTcpci if the system doesn't implement the PD state machines in HW/FW and would like to rely on the OS to provide it
- Use the inbox UCSI driver if your HW/FW supports UCSI over ACPI
- Write a UcmCx client driver if the system cannot use UCSI nor UcmTcpci
  - Least preferred option as it requires the most driver work



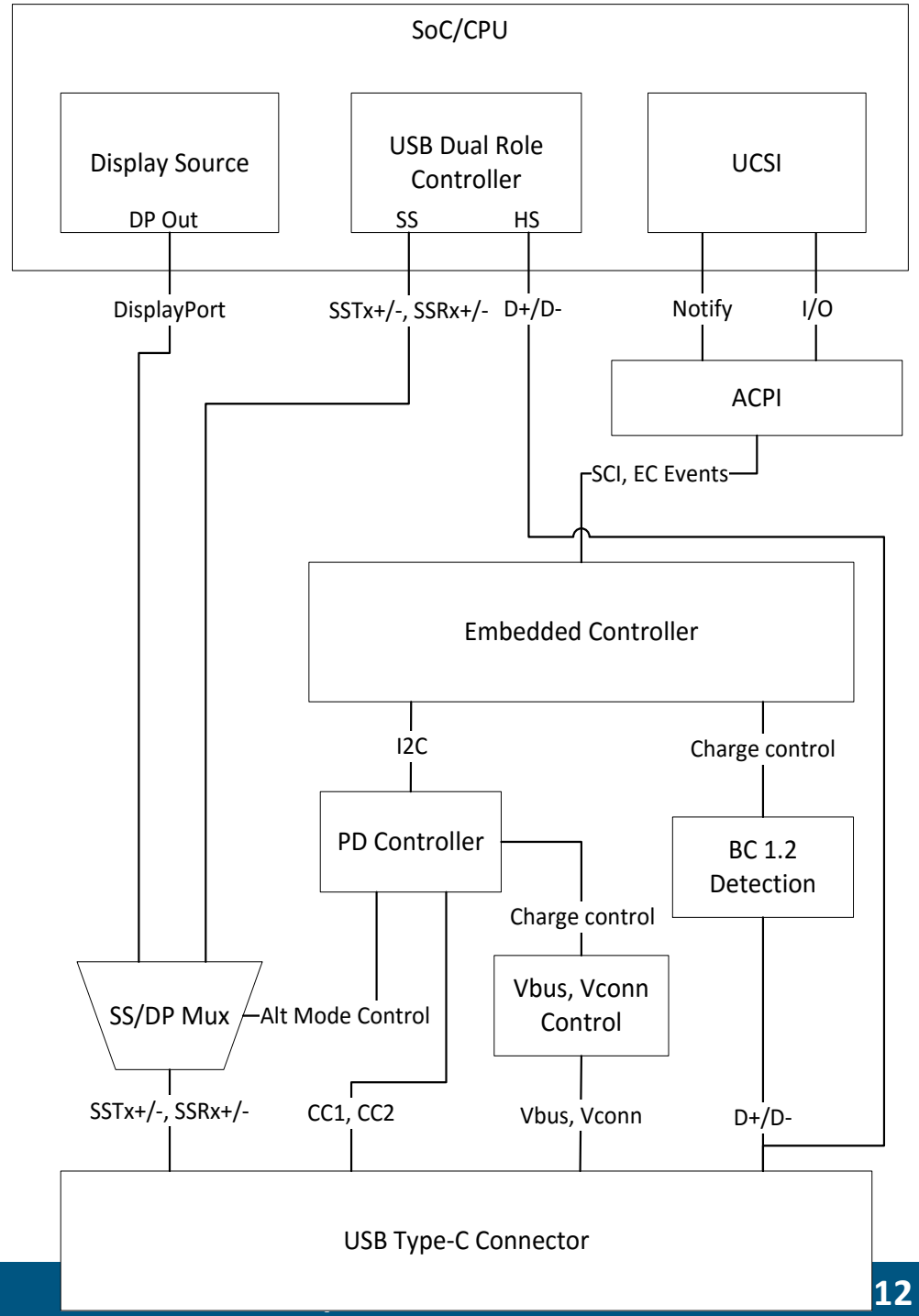
UCSI

- Windows provides an inbox UCSI driver that talks to hardware through ACPI
- Hardware/firmware must implement UCSI specification over ACPI
- System firmware must implement default policies for common expected scenarios.
  - Example: As a sink, always negotiate and charge at the highest available rate
  - Example: DRP devices that are meant to be used in a certain role should always attempt to get into that role



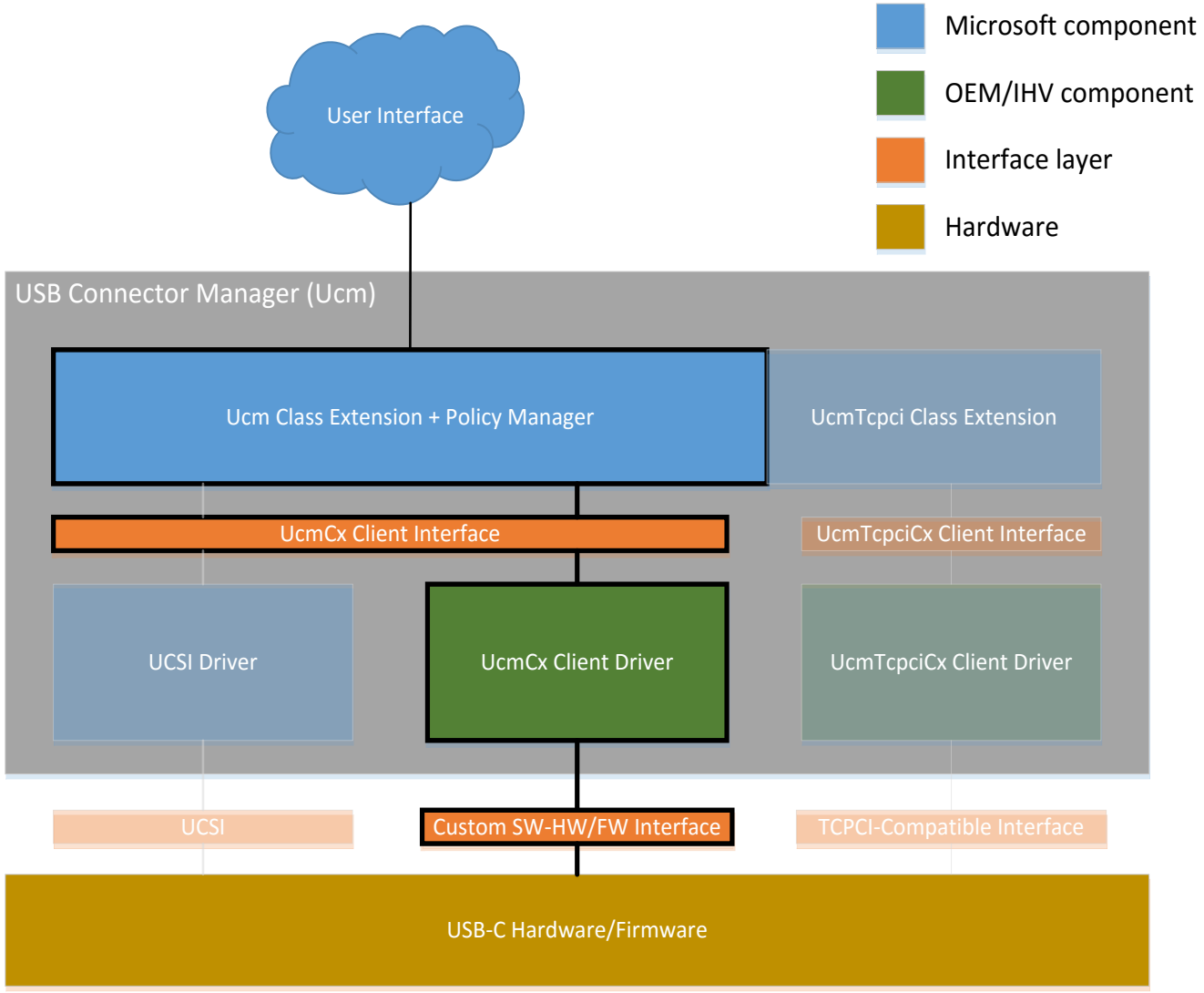
UCSI Bring-up Example

- PD controller tasks:
  - Implement PD policy engine, protocol layer, physical layer
  - Implement default platform policy for the USB-C ports on the system
  - Provide control/query I2C (or other bus) interface hooks for EC
- Embedded controller tasks:
  - Implement I2C (or other bus) interface to PD controller
  - Implement UCSI interface to ACPI
    - Implement SCI, update EC's mailbox
    - Communicate status change, provide role swap hooks, etc
- BIOS ACPI tasks:
  - Report USB-C connector to the OS
  - Implement UCSI mailbox paradigm
    - Initialization of EC's mailbox
    - _DSM method that notifies EC to read the OS message



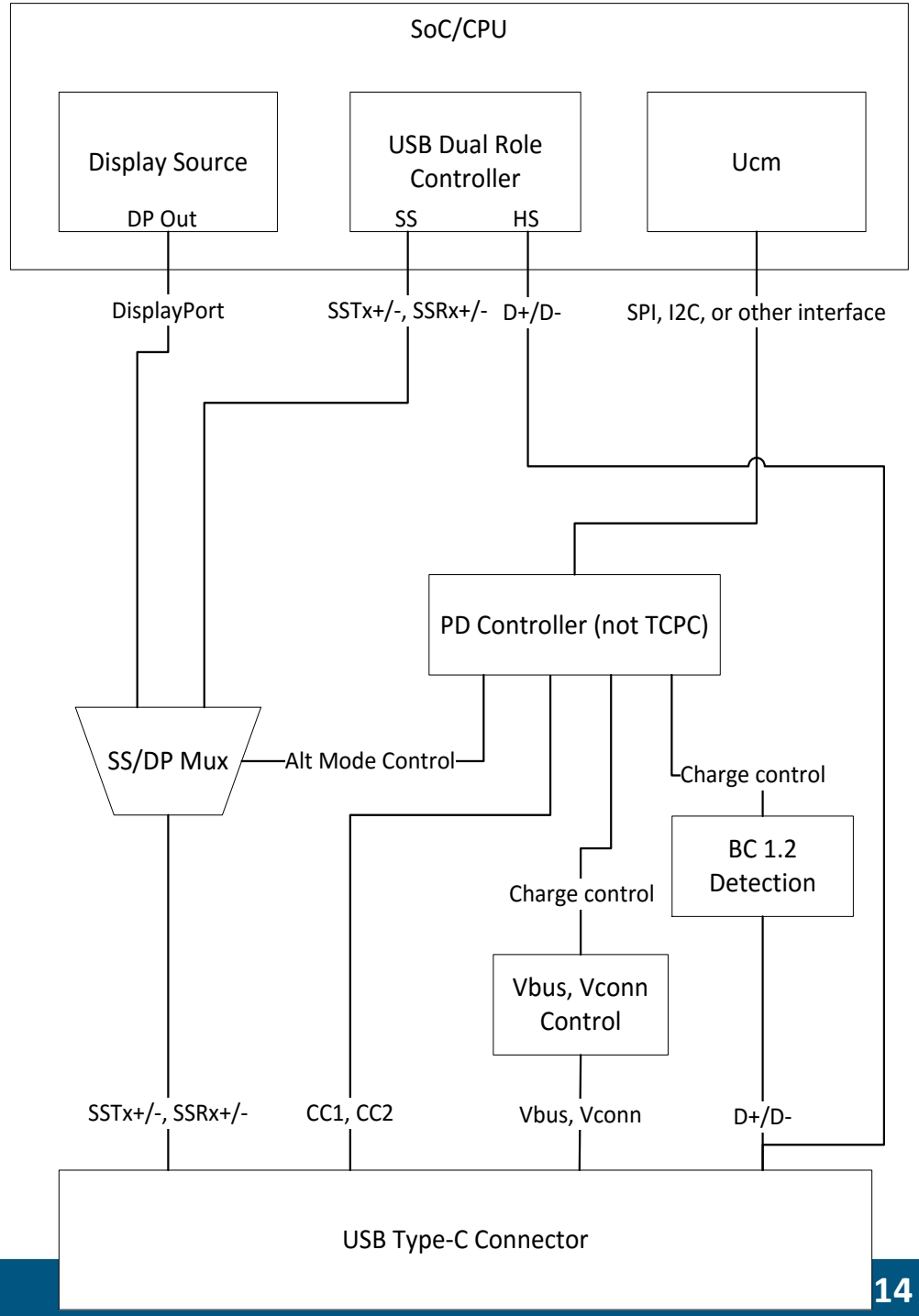
UcmCx Client Driver

- For designs that cannot use Windows' inbox UCSI driver
  - E.g. HW/FW uses a non-UCSI, or non-ACPI interface
- UcmCx client driver is responsible for conveying accurate hardware status to the OS through UcmCx client interface
- UcmCx client driver can use custom interfaces to talk to HW/FW



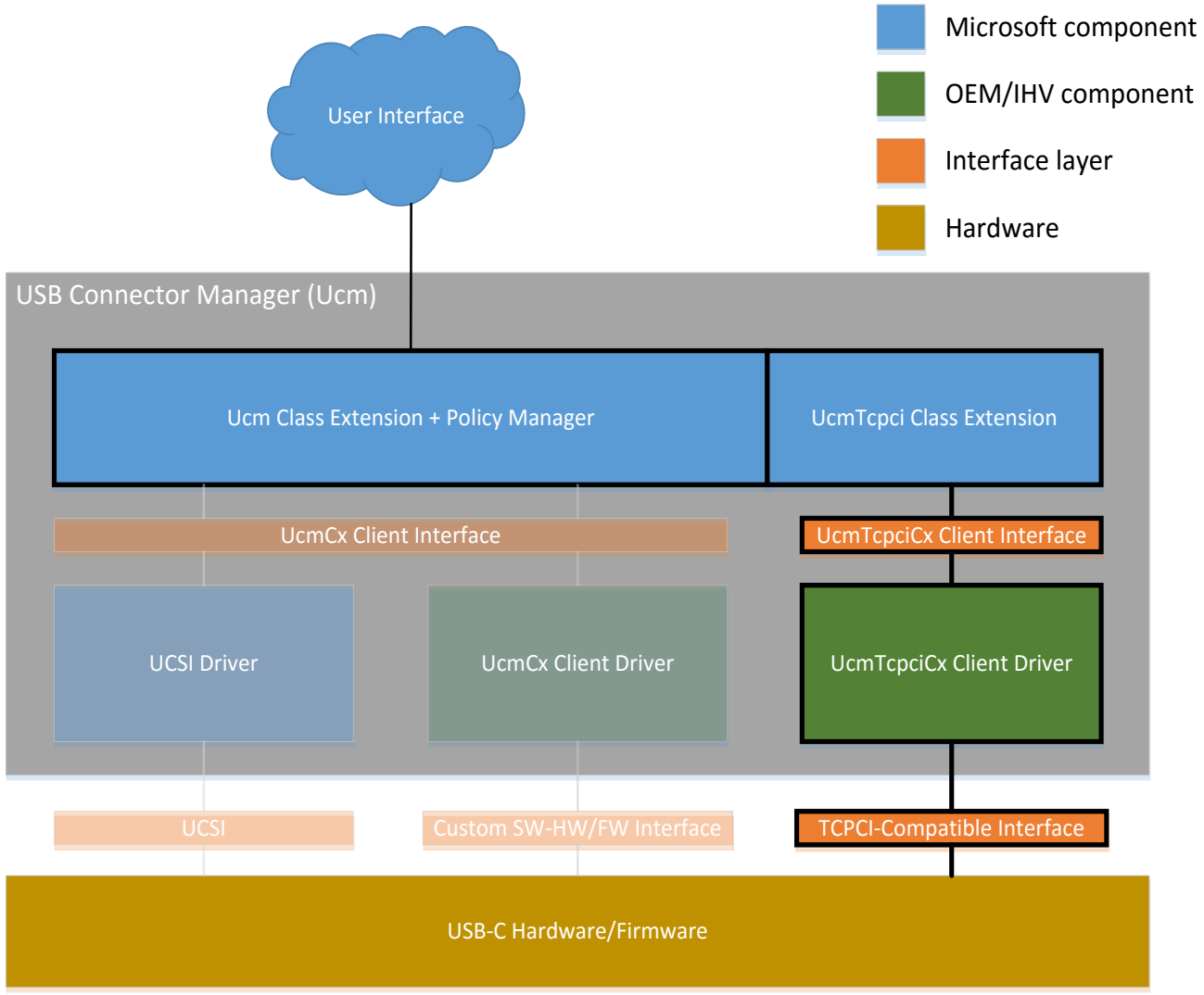
UcmCx Bring-up Example

- PD controller tasks:
  - Implement PD policy engine, protocol layer, physical layer
  - Provide interface hooks for the Ucm client driver
- Ucm client driver tasks:
  - Write an Ucm client driver that implements the Ucm APIs
  - Set default platform policy
  - Implement interface to PD controller
  - Implement UcmCx framework APIs
    - Details on MSDN
- ACPI tasks:
  - Report USB-C connector's capabilities in ACPI



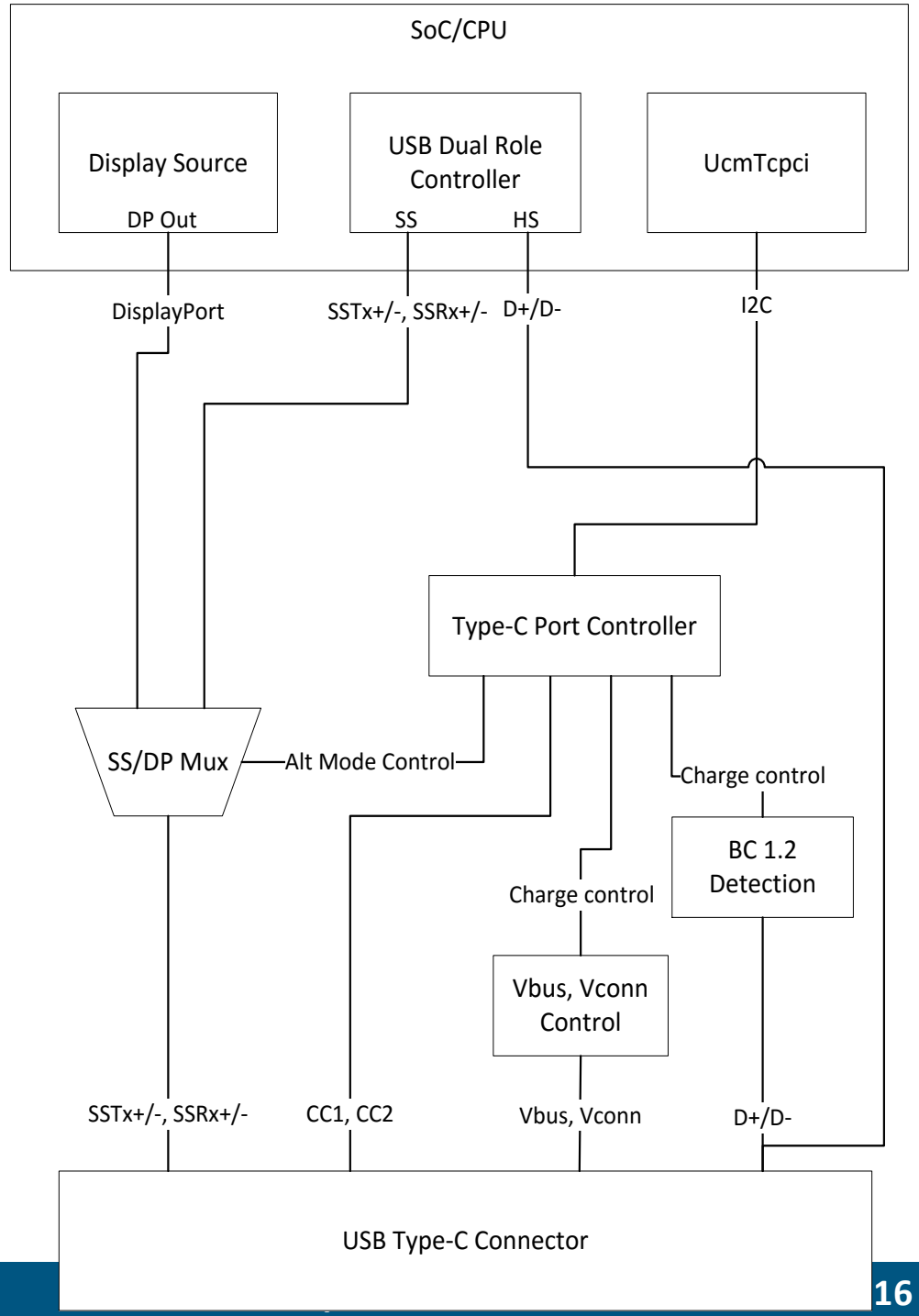
UcmTcpciCx Client Driver

- UcmTcpci is an inbox extension to Ucm
- UcmTcpci is a software-based TCPM implementation
  - Implements PD policy engine and protocol layer
- Designed to be used with platforms with a TCPC, but no HW TCPM
- UcmTcpci client driver can be implemented as a simple I2C class driver
- Driver template available on Windows Drivers Github



UcmTcpciCx Bring-up Example

- Type-C Port Controller tasks:
  - Implement I2C TCPC interface
  - Implement physical layer, CC-logic
- UcmTcpci Client Driver tasks:
  - Write an I2C class driver that implements the UcmTcpci APIs
    - Translate I2C commands to UcmTcpci APIs
    - Sample driver and walkthrough will be available
- ACPI tasks:
  - Report USB-C connector's capabilities in ACPI
- Other tasks:
  - To support PD charging when the system is off, separate charging silicon will be required

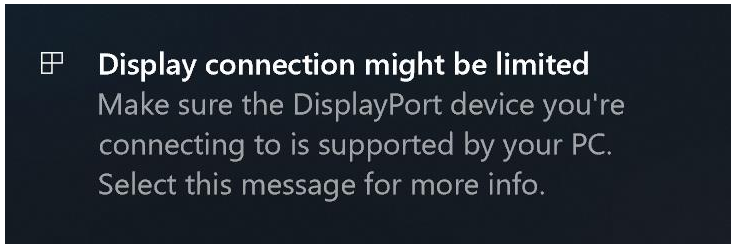


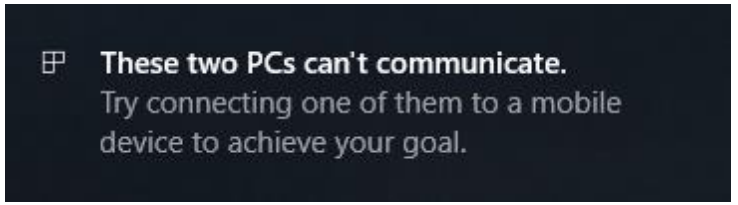
USB-C Troubleshooting Notifications

- List of troubleshooting notifications
- Example sequence diagrams
- Common issues

List of troubleshooting notifications

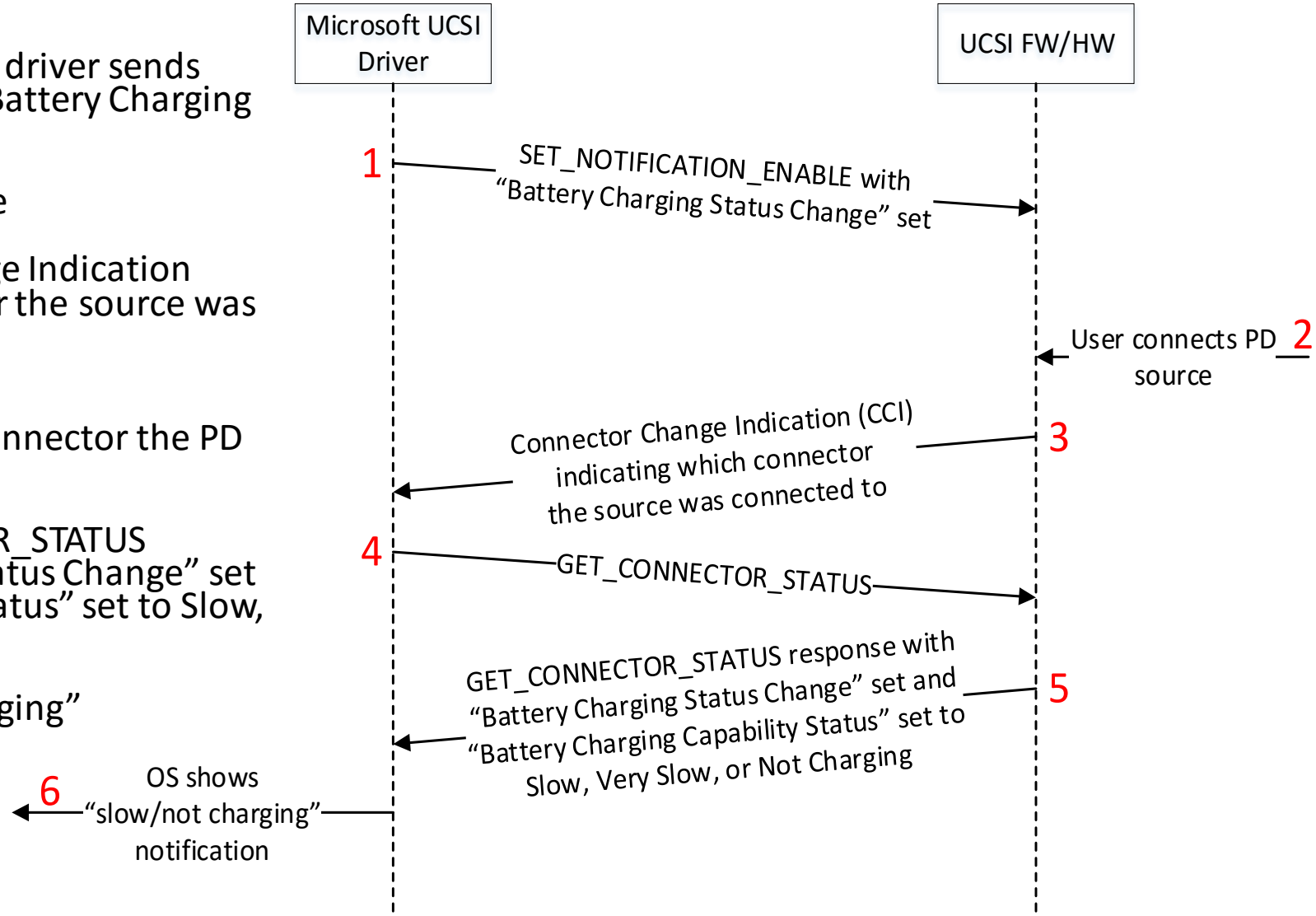
- Charging errors
  - System is charging slowly (or not at all)
- Alternate Mode errors
  - Billboard device
  - Connections to the wrong port on the system
- Device connectivity errors
  - Unsupported Windows-to-Windows connection
  - Connected device declares a power capability mismatch





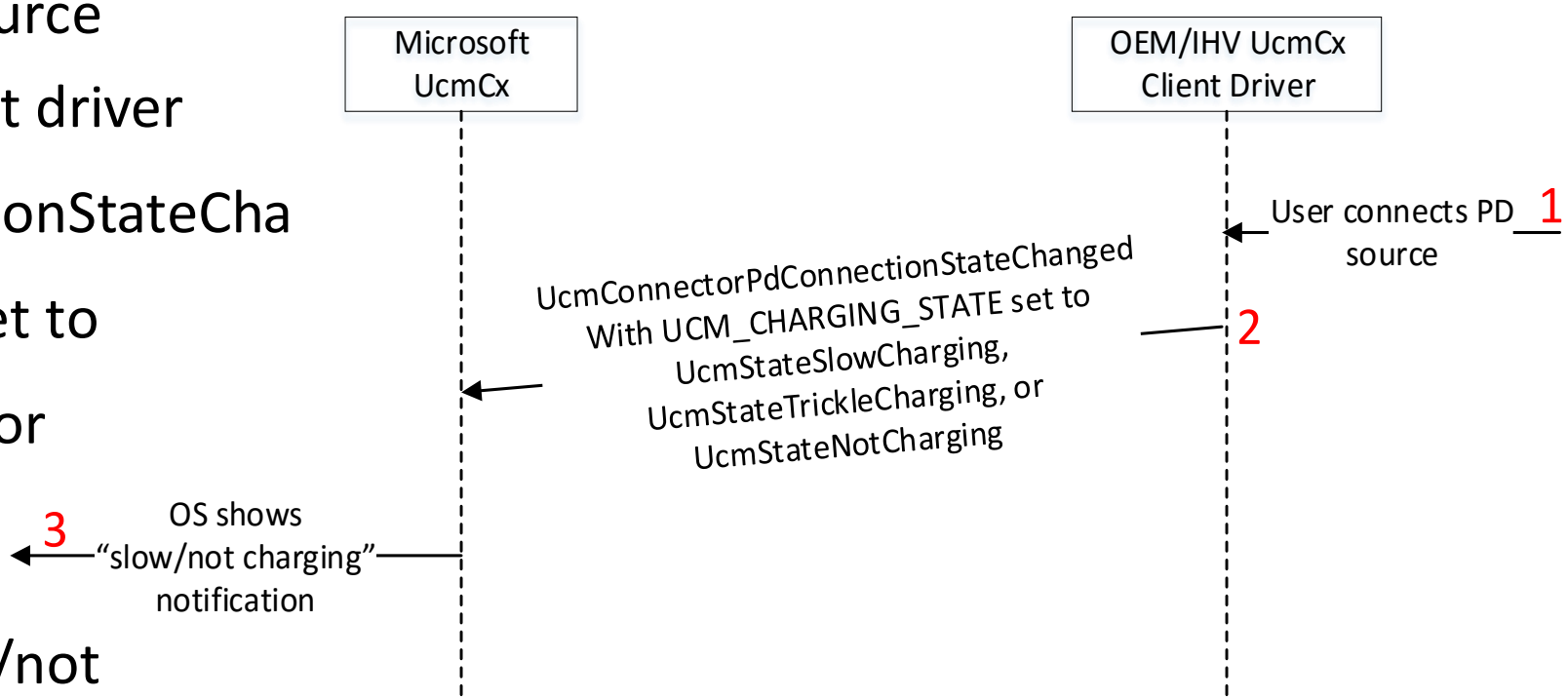
Slow or Not Charging Notification UCSI Sequence Diagram

1. During initialization, Microsoft UCSI driver sends SET_NOTIFICATION_ENABLE with “Battery Charging Status Change” set
2. Later, the user connects a PD source
3. UCSI PPM sends a Connector Change Indication (CCI) that indicates which connector the source was connected to
4. Microsoft UCSI driver sends GET_CONNECTOR_STATUS to the connector the PD source is connected to
5. UCSI PPM returns GET_CONNECTOR_STATUS response with “Battery Charging Status Change” set and “Battery Charging Capability Status” set to Slow, Very Slow, or Not Charging
6. Windows shows the “slow/not charging” notification



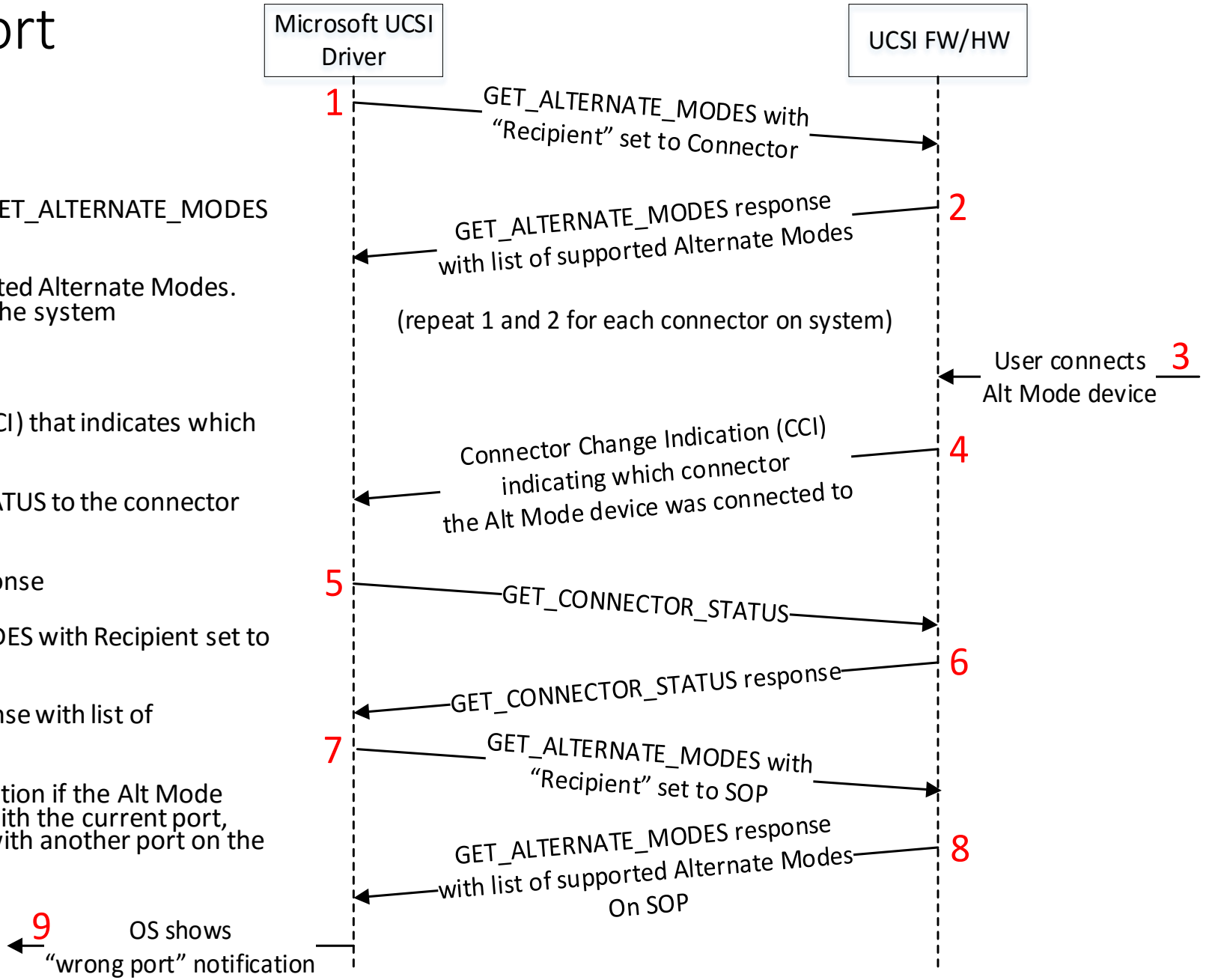
Slow or Not Charging Notification UcmCx Client Driver Sequence Diagram

1. The user connects a PD source
2. The OEM/IHV UcmCx client driver sends UcmConnectorPdConnectionStateChanged command with UCM_CHARGING_STATE set to UcmStateSlowCharging, UcmStateTrickleCharging, or UcmStateNotCharging
3. Windows shows the “slow/not charging” notification



Alternate Mode Wrong Port UCSI Sequence Diagram

1. During initialization, Microsoft UCSI driver sends GET_ALTERNATE_MODES with Recipient set to Connector to UCSI PPM
2. The UCSI PPM returns response with list of supported Alternate Modes. Steps 1 and 2 are repeated for each connector on the system
3. Later, the user connects an Alt Mode device
4. UCSI PPM sends a Connector Change Indication (CCI) that indicates which connector the source was connected to
5. Microsoft UCSI driver sends GET_CONNECTOR_STATUS to the connector the Alt Mode device is connected to
6. UCSI PPM returns GET_CONNECTOR_STATUS response
7. Microsoft UCSI driver sends GET_ALTERNATE_MODES with Recipient set to SOP
8. UCSI PPM returns GET_ALTERNATE_MODES response with list of supported Alternate Modes supported by SOP
9. OS shows “Alternate Mode on wrong port” notification if the Alt Mode device does not share any Alt Modes in common with the current port, but does share at least one Alt Mode in common with another port on the system



Common implementation issues

- "Not charging" or "Slow charging" notifications that don't show up when connected to a slow charger:
  - For UCSI, make sure to set both the "Battery Charging Status Change" bit in addition to the "Battery Charging Capability Status" bits.
  - For UcmCx client drivers, make sure to return UcmConnectorPdConnectionStateChanged notification with "UCM_CHARGING_STATE" field set to the appropriate charging state
- "Not charging" or "Slow charging" notifications that show up when connected to a known good charger:
  - "Battery Charging Capability Status" (UCSI) and "UCM_CHARGING_STATE" (UcmCx client driver) should always reflect the capabilities of the negotiated charging rate
  - Do not set "Slow charging rate" or "Not charging" when connected to non-chargers
  - Example: When connected to a known good charger, always report "Nominal Charger", even if the battery has already been fully charged and the system has stopped charging

USB-C and PD Testing

- Testing guidance
- Requirements and Tests for Win10 Certification
- Win10 Certification Tips and Common Issues

Testing guidance

- Reliable and interoperable hardware results in happy customers
- Microsoft's portal for USB testing resources:
  - https://aka.ms/usbtesting
    - Test tools
    - Testing procedures
    - Test documentation
- Troubleshooting bring-up and development issues:
  - https://aka.ms/usblogs
  - https://aka.ms/usbfeedback
  - https://aka.ms/fieldmedicmsdn
- Give us feedback on how we can better enable you to build high-quality, interoperable hardware!

USB Test tools

- SuperMUTT
  - Simulates USB traffic for USB Host testing
  - USB 2.0 and USB 3.0
- USB Type-C connection exerciser
  - 1:4 switch for automating interop and scenario validation
  - Enables automated switching between hosts, chargers, peripherals
  - Monitors charging current and direction
- USB Type-C SuperMUTT (just released)
  - Extends capabilities of SuperMUTT with USB-C
  - Adds Power Delivery, Alternate Modes, role swapping
  - Tests both System and Charger

USB Type-C SuperMUTT

- Supports all the functionality of the original SuperMUTT
- Adds Type-C and PD support
  - Power Sink/Source (up to 100W)
  - DisplayPort Alt Mode Sink
  - USB Data DFP/UFP
  - Role Swaps
- Reconfigurable to emulate various PD partner types
- Can test both systems and chargers
- Will replace existing Windows HLK tests that require complicated, multi-system setup

Workshops and Certification Programs

- Interop guidance
  - Recommend following the manual interop procedure on MSDN:
    - https://aka.ms/usbtypecmanual
  - Attend interop events hosted by the standards bodies (e.g. USB-IF, VESA, etc)
- Industry Compliance
  - Attend Power Delivery and USB-C compliance workshops hosted by the standards bodies (e.g. USB-IF, VESA, etc)
- Windows Hardware Compatibility Certification
  - New USB-C/PD HLK tests are required for Windows hardware compatibility certification

Requirements and Tests for Win10 Certification

- USB-C/PD HLK Requirements:
  - All USB-C ports are correctly declared in ACPI in _UPC method
  - All required UCSI features must be implemented correctly. See UCSI specification for the list of UCSI features
    - Win10 also requires several UCSI features that are marked as “Optional” in the UCSI specification
  - Ucm APIs must be implemented correctly. See MSDN for the list of APIs and guidance for how to implement those APIs
- USB-C/PD HLK Tests:
  - USB-C ACPI Validation Test
  - UCSI Command Tests
  - Power, Data Role Swap Tests
  - Battery Charging Tests

Win10 Certification Tips and Common Issues

- Always run the latest version of the HLK, regardless of the OS version running on the system under test
- HLK requires both USB-C Connection Exerciser and SuperMUTT
- Systems with a single USB port require the Connection Exerciser's DTMF add-on
- Although not required, Battery Charging HLK tests are highly recommended, especially for platforms with batteries
- Power Role swap tests require the partner to be a DRP
- Detailed HLK setup instructions can be found here: https://aka.ms/usbctestsetup

Future USB-C/PD Support?

Potential Future Features

- UCSI support for non-ACPI transports
  - Microsoft UCSI driver currently supports UCSI over ACPI, but not other transports like PCI, USB, I2C
  - Examples: PCI add-in cards, docks that support Type-C Bridge, PD controllers that support UCSI natively
- TCPC Rev 1.0 v1.2 support in UcmTcpciCx
- Additional troubleshooting notifications
- Peer-to-peer scenarios
  - Screen mirroring, projection
  - File transfer

Q&A

Additional information

- General USB Type-C Information:
  - Windows support for USB Type-C connectors
    - https://aka.ms/usbtypec
- UCSI:
  - UCSI rev1.0 specification
    - http://www.intel.com/content/www/us/en/io/universal-serial-bus/usb-type-c-ucsi-spec.html
  - MSDN UCSI guidance
    - https://aka.ms/ucsi
  - UCSI BIOS enabling guide
    - http://www.intel.com/content/www/us/en/io/universal-serial-bus/bios-implementation-of-ucsi.html
- USB Connector Manager (Ucm):
  - USB Connector Manager Driver bring-up guidance
    - https://aka.ms/ucmcx
  - USB Connector Manager Class Extension API reference
    - https://aka.ms/ucmcxapi

Additional information

- Sample Drivers/Driver Templates
  - UcmCx Client Driver:
    - https://github.com/Microsoft/Windows-driver-samples/tree/master/usb/UcmCxUcsi
  - UcmTcpciCx Client Driver:
    - https://github.com/Microsoft/Windows-driver-samples/tree/master/usb/UcmTcpciCxClientSample
- Win10 Hardware Compatibility Requirements:
  - https://aka.ms/usbccontrolcompat
  - https://aka.ms/usbcsystemcompat
- Win10 Hardware Compatibility Tests:
  - How to set up and run the Win10 USB-C compatibility tests:
    - https://aka.ms/usbctestsetup
- USB testing resources:
  - https://aka.ms/usbtesting
- Test tool questions?
  - MUTTSupport@microsoft.com