



# Certified Wireless USB Framework

**Abdul R. Ismail**  
Intel Corporation

*Content also provided by:  
John S. Howard, Intel Corporation*

# Agenda

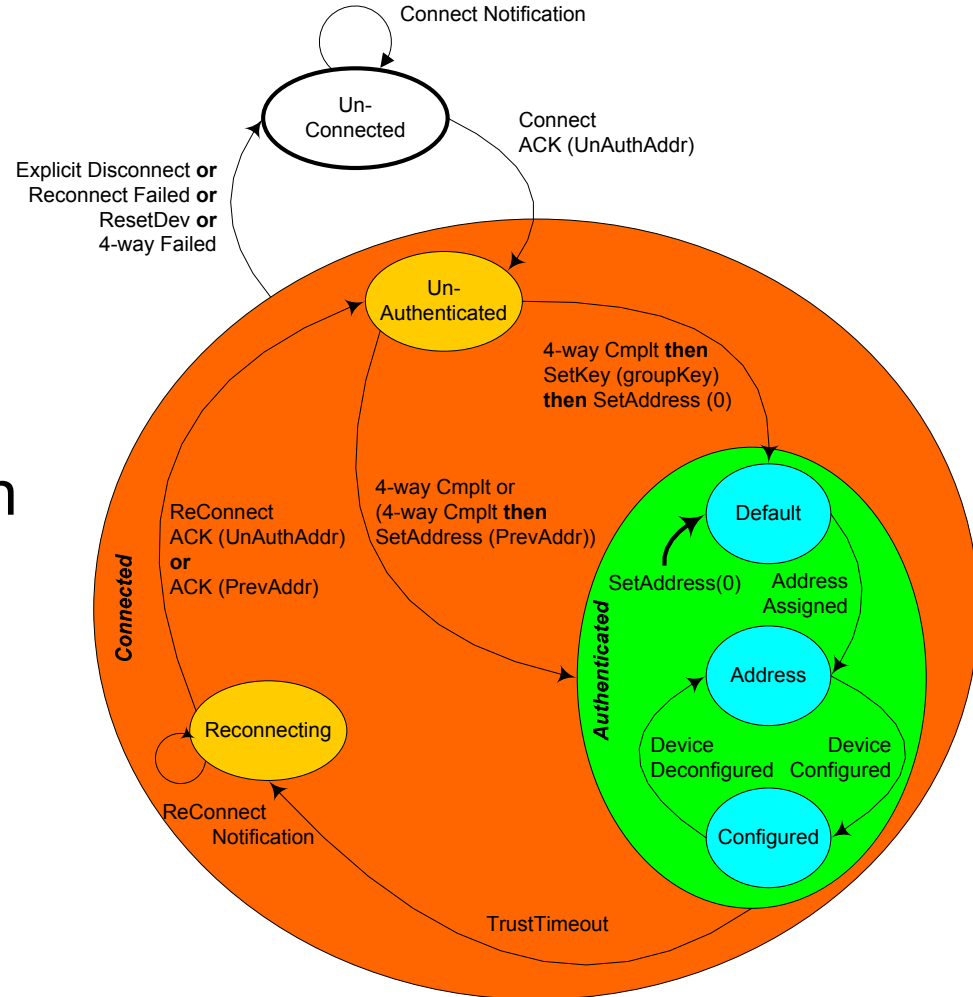


- Device States
- Basic Framework
- New for Certified Wireless USB
  - Summary of Device Requests
  - Device Requests and DBDs
  - Summary of Framework Descriptors
  - Summary of WUSB Channel Information Elements
  - Summary of Device Notifications

# Certified Wireless USB Device States



- USB 2.0 Device state model is preserved within CWUSB Framework
  - Encapsulated within the Secure Connection
- Additional states for establishment and re-establishment of a Secure Connection



# Basic Framework



- USB 2.0 Standard Requests and Descriptors are utilized in Certified Wireless USB
- Restrictions on when requests are valid, based on whether the secure connection is in place
  - No requests that modify device information/state are allowed (except for the requests to establish the secure connection)
- New requests and descriptors added to support Certified Wireless USB requirements
  - Extensions done in a way that enables future extensibility

# Basic Framework

## Summary



<b>Device States</b>	USB 2.0 Encapsulated in Secure Connection	
<b>Requests</b>	Modified	Set Address, Clear Feature, Set Feature, Get Status
	New	Loopback_data_read, Loopback_data_write, Security related requests, SetWSBData
<b>Descriptors</b>	Modified	Device, Configuration, Endpoint
	New	Security, Key, Encryption Type, BOS, Device Capability, Wireless Endpoint Companion
<b>MMC IEs</b>	All New	
<b>Device Notifications</b>	All New	



# Agenda

- Device States
- Basic Framework
- **New for Certified Wireless USB**
  - **Summary of Device Requests**
  - Device Requests and DBDs
  - Summary of Framework Descriptors
  - Summary of WUSB Channel Information Elements
  - Summary of Device Notifications

# Extensions for Device Requests

## Set/Clear Feature



USB 2.0	<b>bmRequest</b> X0000000B	<b>bRequest</b> Set/Clear Feature	<b>wValue</b> Feature Selector	<b>wIndex</b> Zero	<b>wLength</b> zero	<b>Data</b>
------------	-------------------------------	---	--------------------------------------	-----------------------	------------------------	-------------

Certified Wireless USB	<b>bmRequest</b> X0000000B	<b>bRequest</b> Set/Clear Feature	<b>wValue</b> WUSB_Device	<b>wIndex</b> WUSB Feature Selector	<b>wLength</b> zero	<b>Data</b>
------------------------------	-------------------------------	---	------------------------------	---	------------------------	-------------

- Set/Clear Feature
  - Backward compatible with USB 2.0
  - Isolates all CWUSB-specific features from USB 2.0 features
- New standard feature selector: WUSB\_Device
- *wIndex* used for individual WUSB feature selectors

TX DRP IE	Begin transmitting provided DRP IE
DEV XMIT_PACKET	Begin transmitting Transmit Packet
COUNT_PACKETS	Begin counting packets based on parameters provided
CAPTURE_PACKETS	Begin capturing packets base on parameters provided

# Extensions for Device Requests

## Get Status



USB 2.0	<b>bmRequest</b> 1000000B	<b>bRequest</b> Get Status	<b>wValue</b> Zero	<b>wIndex</b> Zero	<b>wLength</b> Two	<b>Data</b> Status
------------	------------------------------	----------------------------------	-----------------------	-----------------------	-----------------------	-----------------------

Certified Wireless USB	<b>bmRequest</b> 1000000B	<b>bRequest</b> Get Status	<b>wValue</b> Zero	<b>wIndex</b> Status Selector	<b>wLength</b> Variable	<b>Data</b> Selector Status Data
------------------------------	------------------------------	----------------------------------	-----------------------	----------------------------------	----------------------------	--

- **Get Status**
  - Backward compatible with USB 2.0
  - Isolates all WUSB-specific Status Information
- *wIndex* used to select different status information

USB 2.0 Status	USB 2.0 Status Information
CWUSB Features	Current setting for CWUSB feature selectors
Channel Info	LQI with respect to last MMC
Received data	Data from Capture/Count Packet operation
MAS Availability	WiMedia MAS Availability bit map
Current Transmit Power	TPC settings for device-level managed packets (Notifications & Beacons)

# New Device Requests

## SetWUSBData



<b>bmRequest</b> 00000000B	<b>bRequest</b> SetWUSBData	<b>wValue</b> WUSB Data Selector	<b>wIndex</b> Zero	<b>wLength</b> WUSB Data Length	<b>Data</b> WUSB Data
-------------------------------	--------------------------------	--	-----------------------	---------------------------------------	--------------------------

- **SetWUSBData**
  - Used to set/update CWUSB-specific descriptors and control parameters
- *wValue* used for individual WUSB data selectors

DPRIE INFO	DRPIE to be used by SBDs for CWUSB Cluster
Transmit Data	Data packet to be transmitted by DBD for TX Packet feature
Transmit Params	DBD TX Packet control parameters
Receive Params	DBS RX count/capture packet parameters
Transmit Power	Host directed control for TX power for certain packet types



# Agenda

- Device States
- Basic Framework
- **New for Certified Wireless USB**
  - Summary of Device Requests
  - **Device Requests and DBDs**
  - Summary of Framework Descriptors
  - Summary of WUSB Channel Information Elements
  - Summary of Device Notifications

# Directed Beaconsing Devices (DBD) Three Functions



- Transmit a packet at certain times (**TX Packet**)
  - This is how the device transmits a beacon
- Count packets at specified time (**Count Packets**)
  - This is how a device can do a beacon or channel scan
  - Device needs to record when packets were received
  - Also grab a few bytes from MAC header
    - Helps host quickly identify beacons that device sees but host doesn't see
- Capture packet at specified time (**Capture Packets**)
  - This is how a device captures neighbor beacons
    - Useful for gathering DRP info
  - Device needs to record when packet was received

# Directed Beaconsing Device

## TX Packet – Things Device Needs to Know



- What to transmit
  - Packet contents are sent to device through control pipe
  - SetWUSBData(TransmitData)
    - 200 bytes of buffer required
- When to transmit
  - Host provides device with two values
    - Transmit time (covers 64K milliseconds)
    - Transmit Adjustment (number of usecs to add each period)
  - SetWUSBData(TransmitParams)
- To transmit or not
  - New Feature Selector: DEV\_XMIT\_PACKET
    - State reflected in generic GetStatus

# Directed Beaconsing Device

## TX Packet – Device Behavior



- Device keeps three values:
  - USB Channel Time: synchronized to time in MMCs
  - Transmit Time: 16 bits, set by host
  - Transmit Adjustment: 4 bits, set by host
- Device Algorithm when enabled:
  - When bottom 16 bits of USB Channel time matches Transmit Time, device sends packet
  - Device adds Transmit Adjustment to Transmit Time
  - Go to 1
- Host will adjust transmit adjustment 'on the fly'
  - Information Element in an MMC
  - Adjustment is the same for all DBD's TX Packet

# Directed Beaconing Device

## Count Packets – Basic Operation



- Device is told when to start counting and when to stop counting
- Host tells device to start counting
- Device counts packets, saves a little bit of packet data and records reception time of each
- Host retrieves count info
- Notes:
  - Host should not expect USB channel responses from device while counting packets
  - The host must ensure that the device is not enabled for transmit before enabling device for counting

# Directed Beacon Device

## Count Packets – Start/Stop Times



- Host tells device when to start counting and for how long
  - Start Time is USB channel time (24 bits)
  - Stop Time is USB channel time (24 bits)
  - Packet filter supported
    - “One-hot” decode on packet type
  - Values are sent through Control channel
  - SetWUSBData(ReceiveParams)
- Host tells device to start counting
  - New Feature Selector: COUNT\_PACKETS
    - One shot behavior. When time period has elapsed, device is no longer enabled to count

# Directed Beaconsing Device

## Count Packets – Saving Info



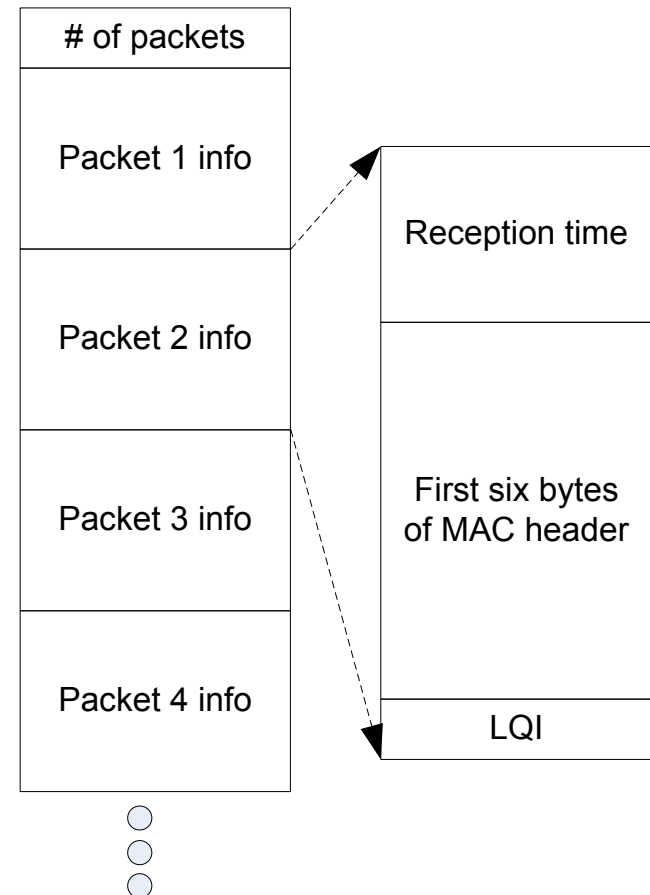
- Device counts packets, saves a little bit of packet data and records reception time of each
  - Data saved is first six bytes of MAC header
    - Packet params, src, destination
  - Device records reception time (USB channel time) when each packet was received
    - Important for locating beacon periods and detecting slow devices
  - LQI recorded also
  - Count buffer must be 512 bytes
    - Each count takes 10 bytes

# Directed Beaconing Device

## Count Packets – Retrieving Data



- Host retrieves count info
  - `GetStatus(ReceivedData)`
  - Device returns data formatted as illustrated



# Directed Beaconsing Device

## Capture Packet – Basic Operation



- Device is told when to start capturing and when to stop
- Host tells device to start capture
- Device captures, records reception time, and saves packet during this time
- Host retrieves captured packet
- Notes:
  - Host should not expect USB channel responses from device while capturing packet
  - The host must ensure that the device transmit time and device capture time don't conflict

# Directed Beaconsing Device

## Capture Packet – Start/Stop Times



- Host tells device when to start capturing and when to stop capturing packets
  - Start is USB channel time
  - Stop is USB channel time
  - Packet filter as well
  - Values are sent through Control channel
  - Set CWUSB Data (Receive Params)
- Host tells device to start capture
  - Feature Selector: Capture Packet
    - One shot behavior. When time period has elapsed, device is no longer enabled to capture

# Directed Beaconsing Device

## Capture Packet – Saving Them



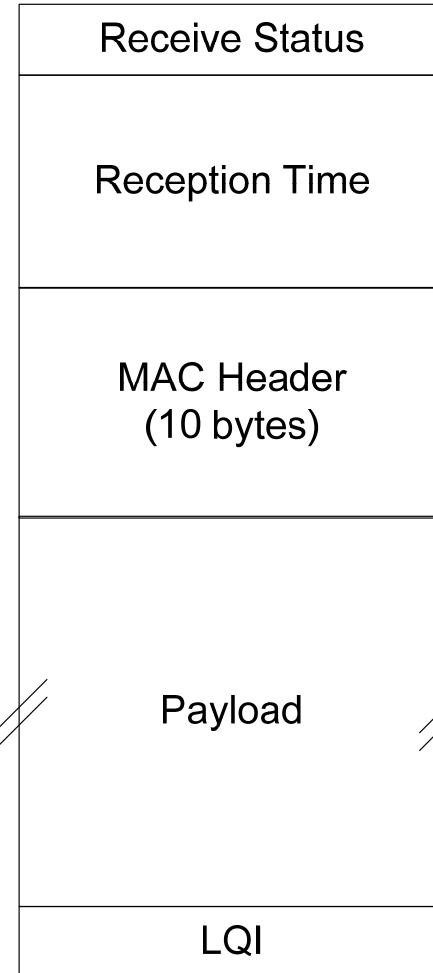
- Device captures, records reception time, and saves packet during this time
  - One packet, 512 bytes of buffering
    - Full packet contents are saved
    - Excess bytes are discarded
  - Device records reception time (USB channel time) when packet was received

# Directed Beaconing Device

## Capture Packets – Retrieving Data



- Host retrieves captured packets
  - New Device Request: Get Status (Received Data)
  - Device returns data formatted as illustrated



Bit 0 set if bad FCS  
Bit 1 set if buffer overrun



# Agenda

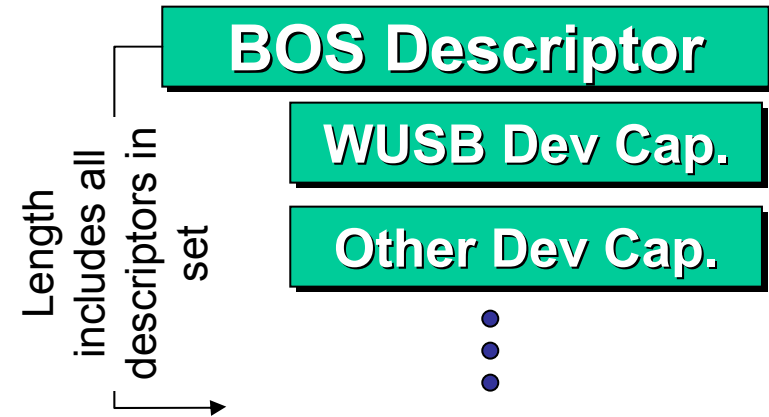
- Device States
- Basic Framework
- **New for Certified Wireless USB**
  - Summary of Device Requests
  - Device Requests and DBDs
  - **Summary of Framework Descriptors**
  - Summary of WUSB Channel Information Elements
  - Summary of Device Notifications

# Modifications for Descriptors

## Device Level



- Device Descriptor
  - bMaxPacketSize0 FFH
  - bcdUSB 0250H
- New BOS Descriptor for Device-level Extensions
  - Accessed via separate request
  - Extensible container model like Configurations
    - Set of descriptors accessed as a set



- CWUSB Device Capabilities
  - P2P – DRD
  - Beacon Behavior
  - TX Rates Supported
  - TX Power Levels Supported
  - Band Groups Supported

# Modifications for Descriptors Configuration Level



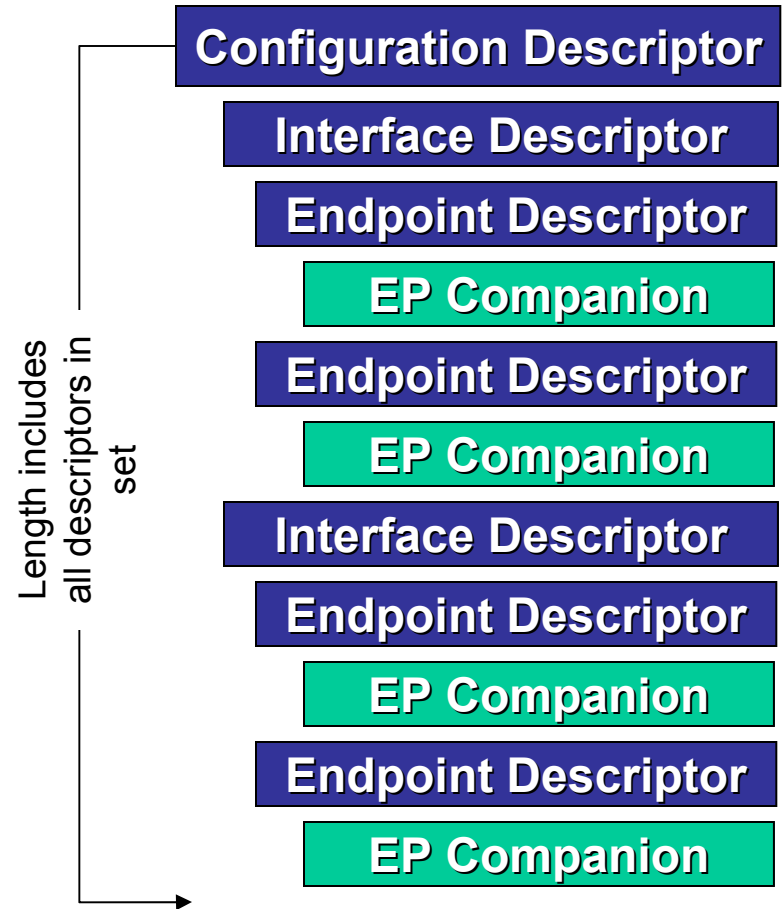
- Configuration Descriptor
  - Required settings for Certified Wireless USB devices
  - New Battery Powered attribute
- Endpoint
  - Changes to:
    - bmAttributes
    - wMaxPacketSize
    - bInterval
  - Backward compatible

# Modifications for Descriptors

## New Endpoint Companion



- Every Endpoint descr. must have an Endpoint Companion descr. in every configuration set
- Endpoint companions descr. must follow the endpoint descr. they are associated with
- Endpoint companions carry new endpoint capabilities for CWUSB

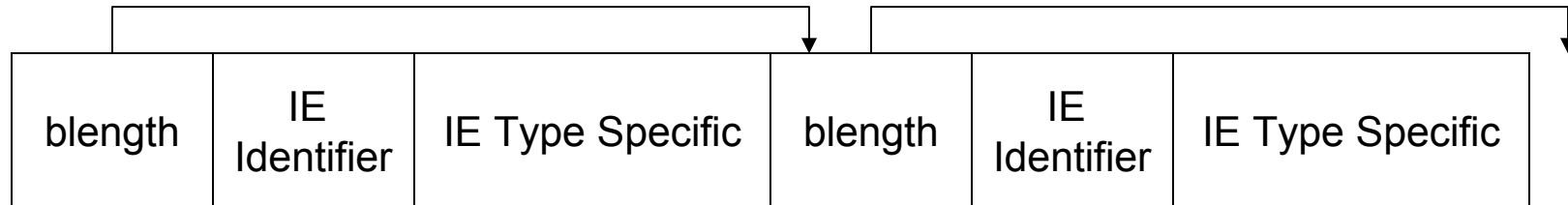




# Agenda

- Device States
- Basic Framework
- **New for Certified Wireless USB**
  - Summary of Device Requests
  - Device Requests and DBDs
  - Summary of Framework Descriptors
  - **Summary of WUSB Channel Information Elements**
  - Summary of Device Notifications

# MMC Information Elements Basics



- *blength* used to find the next IE
- *IE Identifier* determines IE format (type)
- IEs are always an even number of bytes in length
- IEs addressable to multiple devices/endpoints are organized as arrays
  - Most are up to 4 elements (max)
  - Channel Time allocation IE is up to 32 elements (max)
  - May require host to add a pad byte to IE to get it to 2-byte alignment

# MMC Information Elements

## Summary



- Information in MMCs encapsulated in Information Elements

<b>WCTA_IE</b>	Reports transaction group channel time allocations
<b>WCONNECTACK_IE</b>	Host acknowledges connect requests
<b>WHOSTINFO_IE</b>	Host-specific information to uniquely identify Host and the CWUSB Channel
<b>WCHCHANGEANNOUNCE_IE</b>	Information used to announce a channel change time and destination
<b>WDEV_DISCONNECT_IE</b>	Host uses this to disconnect a specific device
<b>WHOST_DISCONNECT_IE</b>	Host uses this to disconnect all devices
<b>WRELEASE_CHANNEL_IE</b>	Host uses this to tell specific device to broadcast a UDR packet (not using channel)
<b>WORK_IE</b>	Host uses this to acknowledge Device sleep requests
<b>WCHANNEL_STOP_IE</b>	Host uses this to announce the channel is stopping
<b>WDEV_KEEPALIVE_IE</b>	Host uses this to force a device to 'check-in'; usually prior to a TrustTimeout
<b>WISOCH_DISCARD_IE</b>	Host uses this on an Isochronous OUT to communicate skips
<b>WRESET_DEVICE_IE</b>	Host uses this as an analog to a USB 2.0 Port Reset signaling
<b>WXMIT_PACKET_ADJUST_IE</b>	Host uses this to communicate xmit adjustments to all devs enabled for Packet Xmit

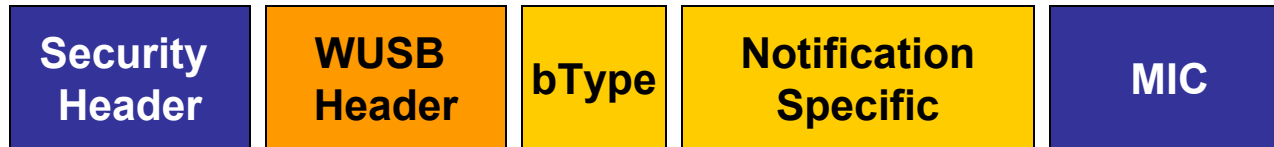


# Agenda

- Device States
- Basic Framework
- **New for Certified Wireless USB**
  - Summary of Device Requests
  - Device Requests and DBDs
  - Summary of Framework Descriptors
  - Summary of WUSB Channel Information Elements
  - **Summary of Device Notifications**

# Device Notifications

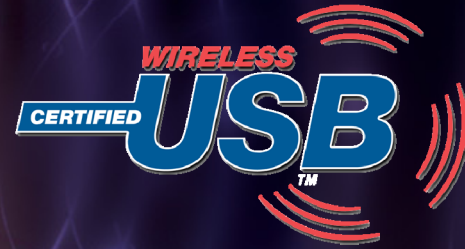
## Basics



- Most Notifications are encapsulated in Secure Packets
- PID in WUSB Header set to DN designates as Device Notification
  - May only be transmitted by devices during Device Notification Time Slots (DNTS CTA)
- *bType* field indicates format of notification-specific payload
- Max notification specific length (payload) is 32 bytes

# Device Notifications

## Summary



<b>DN_Connect</b>	Connect is transmitted without Secure Packet, Reconnect is
<b>DN_Disconnect</b>	Device uses this to explicitly disconnect from Host
<b>DN_EPRdy</b>	Device uses this to report that flow-controlled endpoints are ready to resume data streaming
<b>DN_MASAvailability</b>	Self-beaconing Device uses this to notify that it's MAS availability has changed
<b>DN_RemoteWakeup</b>	Device uses this to emulate USB 2.0 remote-wake signaling
<b>DN_Sleep</b>	Devices uses this to inform the host that it intends to go to a Sleep state
<b>DN_Alive</b>	Device uses this in response to a WDEV_KEEPALIVE_IE or to transition from a sleep state (within a TrustTimeout period)

# Early Prototyping Lessons



- When searching for Host, devices must use the CHID field in Host Info IE (MMC)
  - Ignore HostID and StreamIndex
- DN\_Connect – previous address field must be zero unless doing a ReConnect
- Devices should not expect Host Info IE to be in every MMC
- DNTS CTA may not exist in same MMC as Host Info IE
- Devices (and Hosts) need to observe CTA Start Times, etc.
- Must implement DN\_EPRDY notification for asynchronous communications flow control
  - Non-prototype hosts don't constantly poll
- Devices must be able handle secure frames (as per spec)
- Transmit DN\_Connect wo/secure packet (unless doing a ReConnect)

# Summary



- Extended the USB 2.0 Framework to support Certified Wireless USB
- New device states to support establishing and maintaining a secure connection
  - Basic USB 2.0 defined device state preserved
- New device level capabilities descriptor (extensible)
- New endpoint companion descriptors
- Modified existing endpoint and configuration descriptor for applicability
- Certified Wireless USB device must use bcdUSB value of 0250H
- MMC Information Elements and Device Notifications added to framework



Questions



# Developers Conference 2006

Taipei, Taiwan