Introduction to the Devices Profile for Web Services

Dan Driscoll
Microsoft Corporation
dandris@microsoft.com

September 16, 2008

Agenda

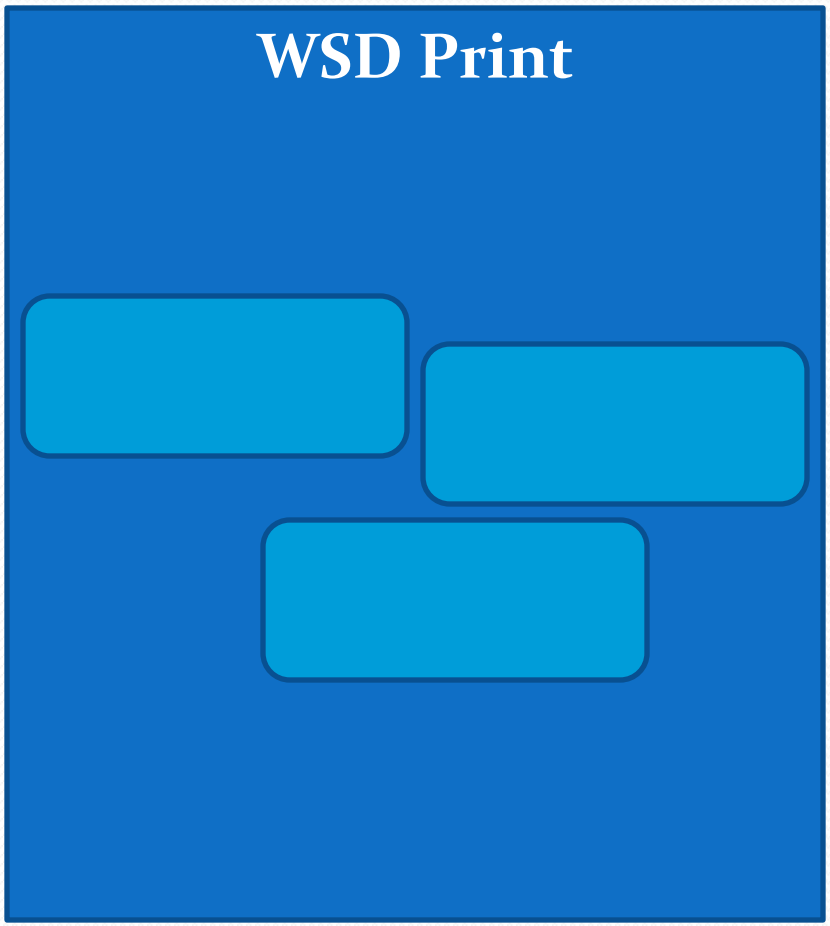
- How WS-* specifications are organized
- Goals of DPWS
- Features and conceptual model
- DPWS from the device perspective

  Devon Kemp, Canon Development Americas, Inc.
- Question and Answer

WS-* specs: building blocks

- Horizontal solutions
- Composable
- Extensible
- Loosely coupled

WS-* specs: solution specs



- Vertical solutions
- Not always composable
- Not always extensible
- Sometimes tightly coupled

WS-* specs: why we need DPWS

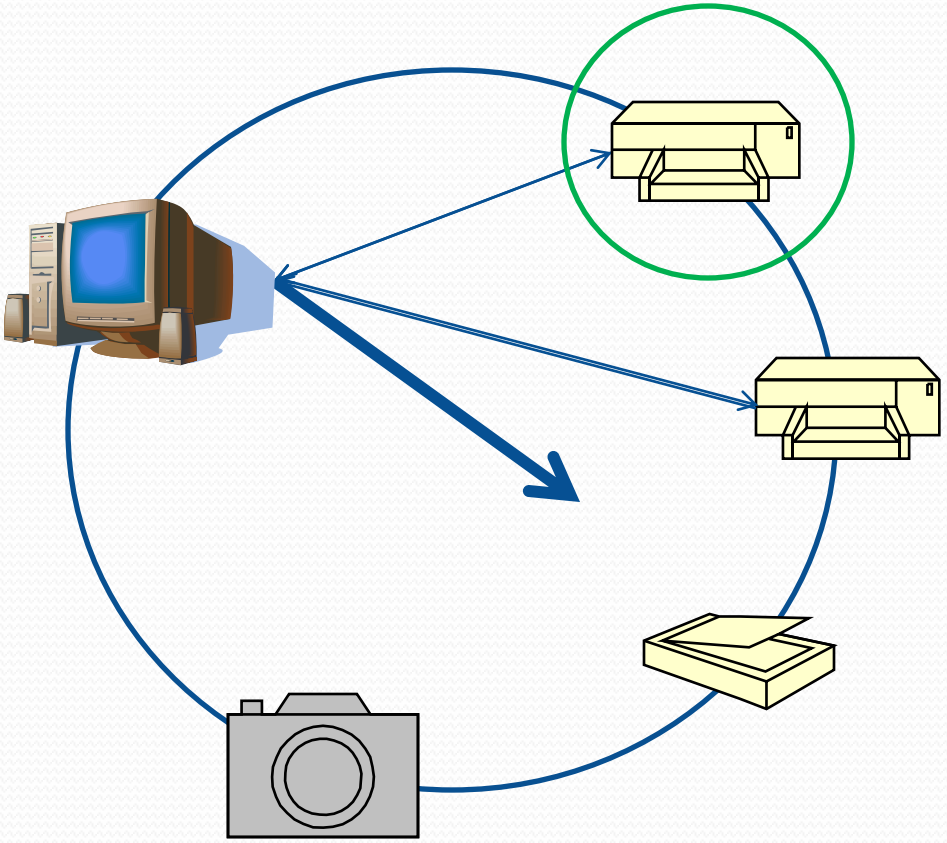
- Uniform choices of building block specifications
- Consistency in constraints applied to building blocks
- Need a generic way to discover and describe devices

Goals of DPWS

- Identify common core specifications
- Constraint specifications so they are easily implemented on devices
- Define functionality common to all devices
- Set a minimum bar for implementation
- Define a conceptual model for devices and clients

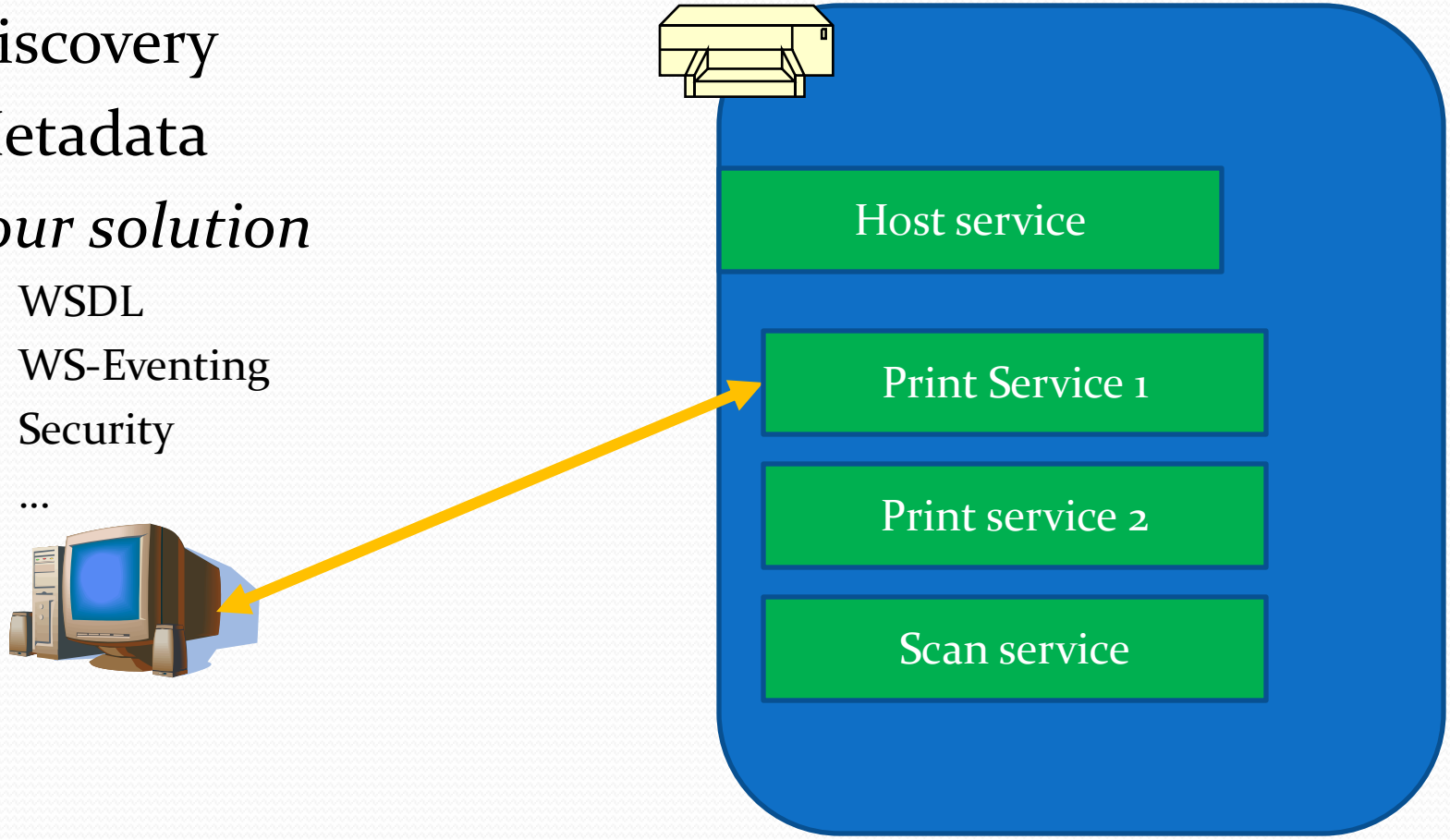
Features: conceptual model

- Discovery
- Metadata
- *Your solution*
  - WSDL
  - WS-Eventing
  - Security
  - …



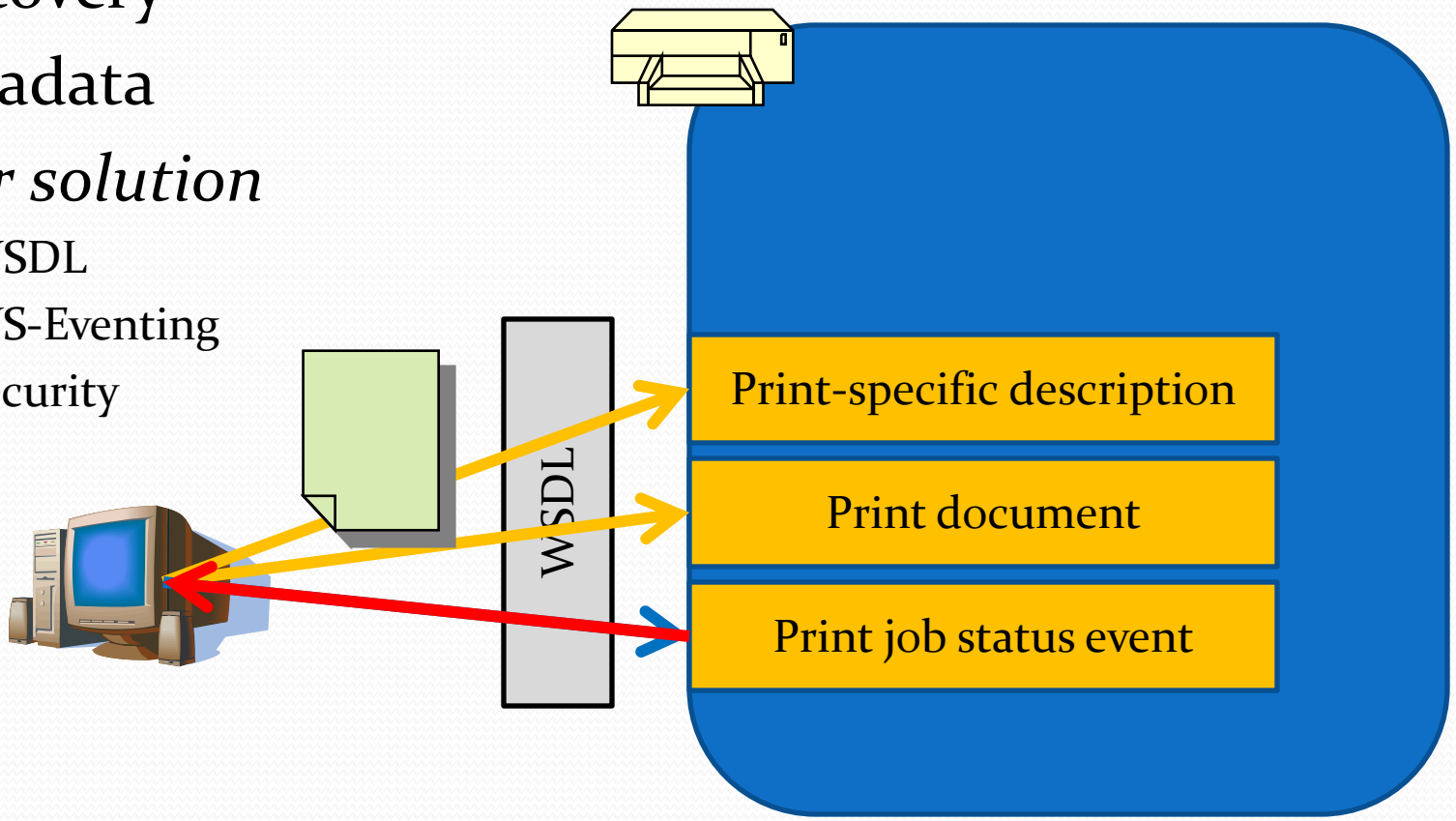
Features: conceptual model

- Discovery
- Metadata
- *Your solution*
  - WSDL
  - WS-Eventing
  - Security
  - …



Features: conceptual model

- Discovery
- Metadata
- *Your solution*
  - WSDL
  - WS-Eventing
  - Security
  - …



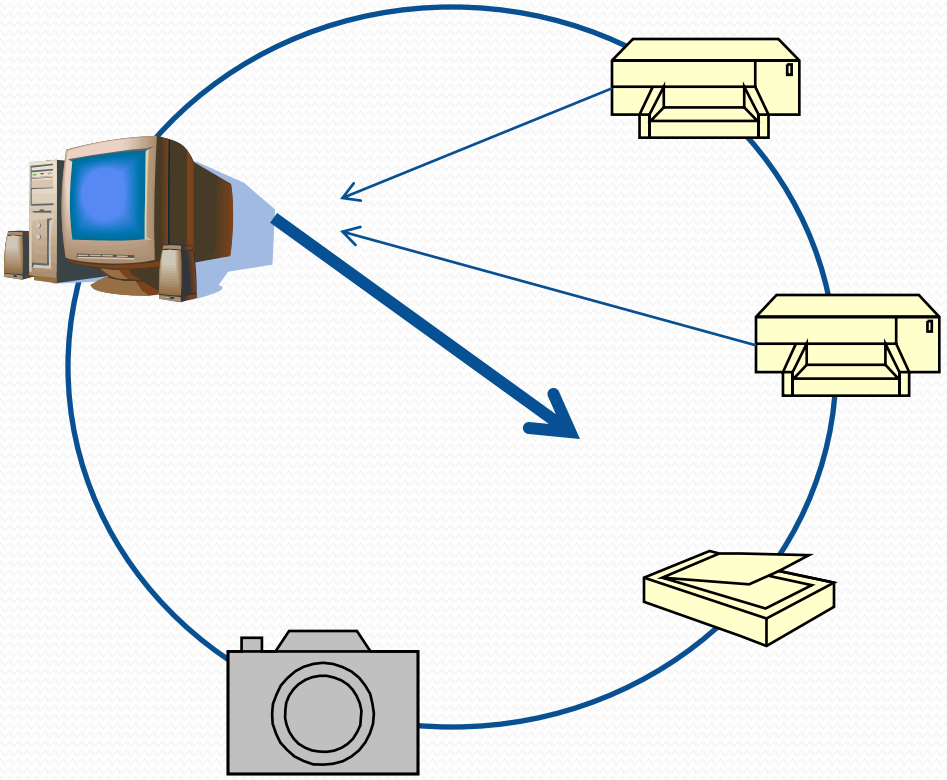
Features: device organization

Every DPWS device has a Host service, and one or more Hosted services:

- Host
  - *dpws:Device* (handles Discovery and Metadata)
- Hosted
  - *Print service* (handles print traffic for printer 1)
  - *Print service* (handles print traffic for printer 2)
  - *Scan service* (handles scan traffic)
  - …

Features: Discovery

- WS-Discovery
- SOAP-over-UDP



Features: Metadata

- WS-Transfer (moves metadata)
- WS-MetadataExchange (defines a container format)
- DPWS Metadata (defines fields within the container)
  - ThisModel: Manufacturer, model name, etc.
  - ThisDevice: Serial number, firmware version, etc.
  - Relationship: Host service, Hosted services
- WSDL (accessible through metadata)

Features: solution-specific

- WSDL
  - Describes a service contract
- WS-Eventing
  - Manages subscription lifetime
- MTOM
  - Moves large binary data out of the SOAP envelope and into a MIME attachment

Features: security

- WS-Discovery compact signatures
- Credentials as x.509 certificates
- Transport layer security (TLS)

A Device Vendor's Experience Implementing DPWS

Sept 18th, 2008

Devon Kemp
Engineering Manager, Canon

The Short Answer

- “it was straight-forward and relatively painless”.
- If you want the “Why?”, keep paying attention…

First…
Why talk about devices?

- Big challenge to add new protocols to devices
  - Highly competitive market
    - Extreme pressure to keep prices low
      - $99 printers
  - Low resources
    - Minimal short-term memory
    - Minimal long-term storage
    - Minimal processing power
    - Energy Star requirements
      - Must use low-energy when not in use.

Usage of common protocols

- Some *Device* Protocols use a variety of *Network* Protocols
  - Sometimes an engineering team needs to learn & implement “one-time” protocols
    - E.g. SSDP, SLP, SNMP Traps,
  - Two costs:
    - Time for the engineering team to learn the new protocol
      - “what is this specification saying??”
    - Device resources consumed by that single-purpose code.
      - E.g. Memory footprint of SSDP can’t be used by anything else.

Usage of common protocols

- DPWS utilizes common protocols
  - As you've already heard…
    - SOAP
    - WSDL
    - MIME (for binary data)
    - Etc.
  - Your engineering team probably already knows the protocols (or at least, they're familiar with them)
    - Engineering "learning curve" is much smaller.
  - Your device may already have these stacks implemented
    - Implementation time is shortened
  - Multiple use code – less resource requirements.
    - Business unit is happy

Uses WSDL

- The services (i.e. Web Interfaces) are defined in WSDL documents
  - Machine readable!
  - Code generating systems exist
  - Doesn't need to be translated to other languages to be understood (like an English spec)

Expandable!

- “Polymorphisizes” easily
  - Easy to be a scanner, printer, fax, etc. or all
    - Just adjust the SOAP messages & events you want to handle .
- Scales vertically easily too
  - Framework has a small footprint, and the services can be defined to have many optional “parts”.
    - e.g. not all events need to be supported
- Easy to use same code base on multiple device ‘types’
  - Ports easily.

DPWS can be ‘quiet’

- Some Device Protocols need to constantly beacon
  - Requires processing power
  - Difficult for device to do other things, such as
    - go to ‘sleep’
    - Process a print job

WSDP Eventing

- Easy to implement, and provides for a rich user experience
  - Reuses the SOAP stack
  - More ‘event’- based, rather than ‘variable’-based.
  - Multiple ‘events’ in one message
    - Allows for a rich eventing mechanism
    - i.e. “Page 3 completed printing”
  - Can approve/deny individual subscription requests
    - Reduces the load of eventing

MTOM

- WSDP uses MTOM/MIME for binary data, and adds certain restrictions
  - The SOAP envelop must be the first MIME part.

- MIME messages are well known
  - Simple to generate / consume for devices.
  - Doesn't require spooling – which devices can't do.

3rd party libraries

- Although Canon does not necessarily use them in production code, numerous 3rd party libraries exist
  - gSOAP, etc.
  - Allows a quick implementation, prototype, or just reference code
  - A WSDP environment can be built quickly for analysis without a large investment.

Questions

Backup slides