This is document is a draft. If you have any suggestions on important capabilities to support or anything else, please let us know. Specifically, we would like feedback on:

- Important capabilities that have been left out
- Complexity or simplicity of the design
- Difficulty in implementation

GCP Device Capability Subsystem

1 Introduction

Printers need a way to describe their capabilities, features and functions in a way that allows a platform to easily generate a UI. In other words, a printer description format needs to simultaneously describe what the printer is capable of, and how those capabilities should be presented to the user. Formats like PPD and XPS are some examples of formats that can express these capabilities. However, a platform like Google Cloud Print (GCP) needs to support printers that express these capabilities in PPD, XPS and others. So we've created a capability format that proposes a simpler and more general solution to this problem: the Cloud Device Description (CDD) format. Of course, this format is not just limited to printers, but can be used for scanners, phones, tablets, web services, or any other device that wants to describe its capabilities in a concise way.

1.1 Design and Other Capabilities Formats

Before CDD, GCP supported the PPD and XPS formats to describe a printer's capabilities. These mature formats have been in use in most major operating systems to handle printing. Each has different ways of expressing capabilities semantically and syntactically. Besides having different file formats, they use different keywords to describe the same printing capabilities. Hierarchy can be expressed explicitly in XPS but not in PPD. Cross-capability constraints are explicitly expressed in PPD as a list of tuplets and XPS implicitly expresses constraints with it's hierarchy structure. XPS's XML format is verbose and not ideal for internet transfer and use on in a web service. PPD's syntax and structure is so flexible that it makes it hard for printers and web services to build parsers for it.

As a web service, GCP had requirements that were not fully supported by the either of the formats. Everytime GCP builds a client UI (e.g. GCP web UI and GCP integration in Chrome's print preview), the client needs to be able to parse all of the keywords in the different PPD and XPS formats in order to present a consistent UI between all printers. Even within a single format, keywords are not used consistently between manufacturers. Beyond our UIs, document providers that integrate with GCP like GMail or Google Drive would have to understand these formats as well in order to understand how a print document should be produced/rendered. Also, the GCP connector would need to understand all of the details of these formats in order to apply transformations to the document before submitting it to the operating system.

The CDD format was created to unify these disparate formats and provide a new simple format for describing printer capabilities with a cloud-first mentality. The CDD format is more concise, stricter, simpler, and built on widely used file formats like JSON and Protobuf. Most importantly, it brings a common semantic understanding to printer capabilities that will make development easier for client UIs, document producers, and cloud devices. It was also built for use by any cloud-connected device, not just cloud printers. This format is expected to grow as novel cloud uses appear.

1.2 Current Limitations of the CDD Format

1. **Expressing capabilities in a hierarchy** - For most common capabilities this is what we want before any additional complexity. This might change in the future.
2. **Cross-capability constraints** - We eventually are going to build this in to prevent user from choosing wrong tray for a given paper type for example.
3. **Defining capability-selections presets** - Users might find it useful to configure several properties at once for "high quality printingin" (like a certain color mode, with a certain DPI setting, etc). We plan on implementing support for this.

2 Cloud Device Description (CDD)

CDD is a format that describes the capabilities of a cloud-connected device, such as a Google Cloud Print connected printer. Some examples of capabilities might include: color printing, duplexing, or multiple paper size support. The format is defined using Google's protobuf language. Here is the top level data structure, CDD:

```
// Description of a cloud-enabled device's capabilities and properties. Also
// known as CDD.
message CloudDeviceDescription {

  // Read-only property that can be used by vendors to further describe the
  // device.
  message VendorProperty {
    required string id = 1;
    required string value = 2;
  }

  // Version of the CDD in the form of "X.Y" where changes to Y are backwards
  // compatible, and changes to X are not.
  required string version = 1;

  // Version of the device's firmware.
  optional string device_firmware_version = 2;

  // URL to direct a user in need of technical support.
  optional string support_url = 3;

  // URL to direct a user to setup your device.
  optional string setup_url = 4;

  // Read-only data that can be used by vendors.
  repeated VendorProperty vendor_property = 100;

  // Special vendor-specific capabilities that are not available in any of the
  // semantic sections of the CDD.
  repeated VendorCapability vendor_capability = 101;

  // Section of the CDD that specifically describes printers.
  optional PrinterDescriptionSection printer = 102;

  // Section of the CDD that specifically describes scanners.
  optional ScannerDescriptionSection scanner = 103;
}
```

Printer capability languages like XPS and PPD specify capabilities using a generic language and then use keywords to designate which of these capabilities have a particular meaning (like which capability refers to the color of the document, or how many copies should be printed). The CDD format takes a different approach by defining a specific set of custom protobufs for each of the device's capabilities.

One can see that the main content of the CDD is broken up into a subsections. Initially, these subsections are *vendor_capability*, *printer*, and *scanner*. From the latter two sections, it can be surmised that the CDD can represent a printer or scanner or even both: a printer-scanner combo. The *vendor_capability* section represents all other capabilities that are currently not supported in any of the other sub-sections.

vendor_capability is a much more free-form section while the *printer* and *scanner* subsections are much more semantic as will be explained below.

2.1 Printer Description Section

This section of the CDD describes the capabilities and features of a printer. Here is a protobuf definition of the supported printer capabilities:

```
// Section of a CDD that describes the capabilities of a cloud-connected
// printer.
message PrinterDescriptionSection {

  // Content types (sometimes referred to as MIME types) that are supported by
  // the printer.
  optional SupportedContentType supported_content_type = 1;

  // Printing speeds that the printer can operate at.
  optional PrintingSpeed printing_speed = 2;

  // PWG raster configuration of the printer. Only set this if the printer
  // supports image/pwg-raster content type. This allows a cloud service to
  // understand how to rasterize a document for the printer.
  optional PwgRasterConfig pwg_raster_config = 3;

  // Color printing capabilities of the printer.
  optional Color color = 100;

  // Duplexing capabilities of the printer.
  optional Duplex duplex = 101;

  // Page/paper orientation capabilities of the printer.
  optional PageOrientation page_orientation = 102;

  // Multiple copy capability of the printer.
  optional Copies copies = 103;

  // Page margins capability of the printer.
  optional Margins margins = 104;

  // Printing quality or dots-per-inch (DPI) capabilities of the printer.
  optional Dpi dpi = 105;

  // Page fitting capabilities of the printer.
  optional FitToPage fit_to_page = 106;

  // Page range selection capability of the printer.
  optional PageRange page_range = 107;

  // Page or media size capabilities of the printer.
  optional MediaSize media_size = 108;

  // Paper collation capability of the printer.
  optional Collate collate = 109;

  // Reverse order printing capability of the printer.
  optional ReverseOrder reverse_order = 110;

  // Automatic page rotation capability of the printer.
```

```
  optional RotateToPage rotate_to_page = 111;
}
```

More capabilities that are common to most printers will eventually be added to this section. If a printer has a particular capability, the CDD author should simply set one of the fields above.

Note that not all of the above fields are capabilities that the user can tweak. Some of them (like *printing_speed*) are merely descriptive, and serve to help cloud services (like GCP) to understand how a job will be handled by the device.

The printer-specific capability proto definitions are given in detail in the Appendix. For example, here is one of these proto definitions for the *duplex* capability:

```
// Capability that defines a set of duplexing options available on a device.
message Duplex {
  enum Type {
    NO_DUPLEX = 0;
    LONG_EDGE = 1;
    SHORT_EDGE = 2;
    MANUAL_DUPLEX = 3;
  }

  message Option {
    required Type type = 1;
    optional bool is_default = 2 [default = false];
  }

  repeated Option option = 1;
}
```

This object would allow a UI for selecting printing capabilities to populate a drop-down list of duplexing options by iterating the options listed in the *duplex* capability.

2.2 Scanner Description Section

The CDD strives to be device-agnostic and so can also describe the capabilities of a scanner. Here is the protobuf definition of the scanner description section of a CDD:

```
// Section of a CDD that describes the capabilities of a cloud-connected
// scanner.
message ScannerDescriptionSection {

  // Color scanning capabilities of the scanner.
  optional Color color = 100;

  // Image quality or dots-per-inch capabilities of the scanner.
  optional Dpi dpi = 102;

  // Image size capabilities of the scanner.
  optional MediaSize media_size = 103;

  // File format type capabilities of the scanner.
  optional FileFormat file_format = 104;
}
```

New capabilities common to most scanners will be added to this protobuf. These existing capabilities are described in detail in the Appendix. For example here's the protobuf specification of the *dpi* capability:

```
// Capability that defines a set of 2D image quality levels available on a
// device.
message Dpi {
  enum Type {
    CUSTOM = 0;
    DRAFT = 1;
    NORMAL = 2;
    PHOTO = 3;
    BEST = 4;
  }

  message Option {
    required Type type = 1;
    required int32 horizontal_dpi = 2;
    required int32 vertical_dpi = 3;
    optional bool is_default = 4 [default = false];
  }

  repeated Option option = 1;
  optional int32 min_horizontal_dpi = 2;
  optional int32 max_horizontal_dpi = 3;
  optional int32 min_vertical_dpi = 4;
  optional int32 max_vertical_dpi = 5;
}
```

Notice that in the *dpi* capability, each option specifies the horizontal and vertical DPI values so that user interfaces can show the user exactly what resolution each of the options represent for example.

2.3 Vendor Capabilities

If a cloud-connected device exhibits capabilities that don't belong into any of the semantic sections of the CDD (printer, scanner, etc.), then an author can make use of the *vendor_capability* repeated field. This field can hold vendor-specific capabilities that come in one of three flavors: range-based, selection-based, and typed-value-based. Here's the protobuf definition of vendor capabilities:

```
// Flexible capability that can represent range-base, selection-based, or
// typed-value based capabilities.
message VendorCapability {
  enum Type {
    RANGE = 0;
    SELECT = 1;
    TYPED_VALUE = 2;
  }

  // ID of the capability. Used in CJT to associate a ticket item with this
  // capability.
  required string id = 1;

  // User-friendly string to represent this capability. This string is not
  // localized.
  required string display_name = 2;
```

```
  // Type of this capability.
  required Type type = 3;

  // Range-based capability definition.
  optional RangeCapability range_cap = 4;

  // Selection-based capability definition.
  optional SelectCapability select_cap = 5;

  // Typed-value-based capability definition.
  optional TypedValueCapability typed_value_cap = 6;
}
```

Vendor capabilities come in different flavors. Currently, three are supported:

- Range based capabilities
- Typed-value based capabilities
- Selection based capabilities

2.3.1 Range Based

A ranged based capability just consists of a value type {integer, float}, a default value of the capability, and a minimum and maximum value of the capability. An example of a range-based capability might be “printed copies”, where the default might be “1”, the minimum “1”, and the maximum whatever value your device supports. Here is the protobuf definition for a range based capability:

```
// Message that stores capability information specific to range-based
// capabilities.
message RangeCapability {
  enum ValueType {
    FLOAT = 0;
    INTEGER = 1;
  }

  required ValueType value_type = 1;
  optional string default = 2;
  optional string min = 3;
  optional string max = 4;
}
```

2.3.2 Typed-value Based

A typed-value based capability is exactly like a range based capability except without the min and max, and with more data types. An example of a typed-value based capability might be a one-time-use access code, or a filename for saving a print-out. Here is the protobuf definition of a typed-value capability:

```
// Message that stores capability information specific to typed-value-based
// capabilities.
message TypedValueCapability {
  enum ValueType {
    BOOLEAN = 0;
    FLOAT = 1;
    INTEGER = 2;
    STRING = 3;
  }
```

```
  required ValueType value_type = 1;
  optional string default = 2;
}
```

2.3.3 Selection Based

Probably one of the most common capabilities (of printers) are the selection based capabilities. These capabilities present a list of predefined options for the user to choose from. An example of a selection based capability might be what kind of duplexing to apply to a print job, or what type of color model to print with. Here is the protobuf definition of a selection based capability:

```
// Selection-based device capability. Allows the user to select one or many of
// a set of options.
message SelectCapability {

  // A user-selectable option of the vendor capability.
  message Option {

    // A single string that represents the value of this option. This value
    // will be used in the VendorTicketItem.value field.
    required string value = 1;

    // User-friendly string to represent this option.
    required string display_name = 2;

    // Whether this option is the default option. Only one option should be
    // marked as default.
    optional bool is_default = 3 [default = false];
  }

  // List of options available for this capability.
  repeated Option option = 1;
}
```

2.4 Examples

Some examples of CDDs might help to understand how CDDs can be used to describe a cloud-connected device. As will be described in section Integration with GCP, the protobufs defined in the CDD specification are transmitted in JSON format. The following illustrative examples are written in JSON.

2.4.1 Typical Printer

Here's an example of what a CDD would look like for a printer that supports color printing, multiple copies, and multiple page sizes and that can support natively printing PDFs, JPEGs, and plain text:

```
{
  "version": "1.0",
  "vendor_property": [],
  "vendor_capability": [],
  "printer": {
    "supported_content_type": {
      "option": [
```

```
        {
          “content_type”: “application/pdf”,
          “rank”: 1,
          “min_version”: “1.5”
        },
        {
          “content_type”: “image/jpeg”,
          “rank”: 2
        },
        {
          “content_type”: “text/plain”,
          “rank”: 3
        },
      ]
    },
    “color”: {
      “option”: [
        {
          “vendor_id”: “grayscale”,
          “type”: 1, // MONOCHROME
        },
        {
          “vendor_id”: “color”,
          “type”: 0, // COLOR
          “is_default”: true
        },
        {
          “vendor_id”: “ultra-color”,
          “type”: 2, // CUSTOM_COLOR
          “custom_display_name”: “Best Color”
        },
      ]
    },
    “copies”: {
      “default”: 1,
      “max”: 511
    },
    “media_size”: {
      “option”: [
        {
          “type”: 6, // A4
          “width_microns”: 210000,
          “height_microns”: 297000,
          “is_default”: true
        },
        {
          “type”: 9, // LEGAL
          “width_microns”: 215900,
          “height_microns”: 355600
        },
        {
          “type”: 8, // LETTER
          “width_microns”: 215900,
          “height_microns”: 279400
        }
      ],
    }
  ],
}
```

2.4.2 File Saving Device

Here is an example of a CDD for a device that receives a print job and saves it in a specified folder and with a specified filename and can only save PDFs:

```
{
  “version”: “1.0”,
  “vendor_property”: [],
  “vendor_capability”: [
    {
      “id”: “folder-path”,
      “display_name”: “Destination Folder”,
      “type”: 2, // TYPED_VALUE
      “typed_value_cap”: {
        “value_type”: 3, // STRING
        “default”: “/tmp/”
      }
    },
    {
      “id”: “filename”,
      “display_name”: “File Name”,
      “type”: 2, // TYPED_VALUE
      “typed_value_cap”: {
        “value_type”: 3, // STRING
        “default”: “printout.pdf”
      }
    }
  ],
  “printer”: {
    “supported_content_type”: {
      “option”: [
        {
          “content_type”: “application/pdf”,
          “rank”: 1
        }
      ]
    }
  }
}
```

3 Cloud Job Ticket (CJT)

After a user makes a selection and configures the values of the device's capabilities to prepare a print job, a Cloud Job Ticket, or "ticket" for short, is constructed to instruct the printer or other device on how to handle the print job. This ticket is sent from the printing client (e.g. mobile phone application) to the cloud service (e.g. GCP) and stored together with the job data. Like the CDD, the CJT is broken up into device-specific sections. For example, there are separate sections of ticket items concerned with printers and those concerned with scanners. Here is the protobuf definition of a CJT:

```
// Description of how a cloud job (e.g. print job, scan job) should be handled
// by the cloud device. Also known as CJT.
message CloudJobTicket {

  // Version of the CJT of the form "X.Y". Where an increment in Y is backwards
  // compatible but an increment in X is not.
  required string version = 1;

  // List of vendor-specific ticket items.
  repeated VendorTicketItem vendor_ticket_item = 2;

  // Section of CJT pertaining to cloud printer ticket items.
  optional PrinterTicketSection printer = 3;

  // Section of CJT pertaining to cloud scanner ticket items.
  optional ScannerTicketSection scanner = 4;
}
```

3.1 Printer Ticket Section

The printer ticket section of the CJT describes how a print job should be handled by a cloud-connected printer. Here is the protobuf definition of the printer ticket section:

```
// Section of a CJT of how a print job should be handled by a cloud-connected
// printer.
message PrinterTicketSection {
  optional ColorTicketItem color = 1;
  optional DuplexTicketItem duplex = 2;
  optional PageOrientationTicketItem page_orientation = 3;
  optional CopiesTicketItem copies = 4;
  optional MarginsTicketItem margins = 5;
  optional DpiTicketItem dpi = 6;
  optional FitToPageTicketItem fit_to_page = 7;
  optional PageRangeTicketItem page_range = 8;
  optional MediaSizeTicketItem media_size = 9;
  optional CollateTicketItem collate = 10;
  optional ReverseOrderTicketItem reverse_order = 11;
}
```

The protobuf definition of each ticket item can be found in the Appendix. Most of the ticket items are simple: containing only one field that identifies the option chosen by the user. If a ticket item is left unset, then the cloud service will use a default value specified in the CDD. Also notice that not all of the fields in the printer

section of the CDD have a corresponding field in the printer section of the CJT (like *printing_speed*). This is because these fields are not modifiable by the user and so don't belong in the CJT. The *margins* ticket item is one of the more complicated ones. It's protobuf definition is reproduced here as an example:

```
// Ticket item indicating what margins to use (in microns).
message MarginsTicketItem {
  required int32 top_microns = 1;
  required int32 right_microns = 2;
  required int32 bottom_microns = 3;
  required int32 left_microns = 4;
}
```

3.2 Scanner Ticket Section

Similar to the printer ticket section, the scanner ticket section describes the ticket items associated with scan jobs. Here is its protobuf definition:

```
// Section of a CJT of how a scan job should be handled by the cloud-connected
// scanner.
message ScannerTicketSection {
  optional ColorTicketItem color = 1;
  optional DpiTicketItem dpi = 2;
  optional MediaSizeTicketItem media_size = 3;
  optional FileTypeTicketItem file_type = 4;
}
```

Notice that the scanner ticket section has a *dpi* field just like the printer section. These fields are independent even though they use the same protobuf. Sharing of protobuf objects might be common between different types of devices. For now, Dpi, Color, and MediaSize are shared between the printer and scanner sections.

3.3 Vendor Ticket Items

Like the *vendor_capability* field of the CDD, the CJT contains ticket items that do not belong to capabilities specified in either the printer or scanner sections (or any other section that will be added later). These ticket_items are much simpler and consist only of an ID and value pair, where the ID refers to a corresponding vendor capability defined in the CDD and the value corresponds to the user input or user selection. Here is the protobuf description of a vendor ticket item:

```
// Ticket item indicating what value for a vendor-specific capability to use.
message VendorTicketItem {

  // ID of vendor-specific capability that this ticket item refers to.
  required string id = 1;

  // Value of ticket item.
  required string value = 2;
}
```

3.4 Examples

Here is an example of what a ticket would look like produced for the “Typical Printer” example above:

```
{
  “version”: “1.0”,
  “vendor_ticket_item”: [],
  “printer”: {
    “color”: {
      “vendor_id”: “grayscale”,
      “type”: 1 /*MONOCHROME*/
    },
    “copies”: { “copies”: 3 }
  }
}
```

Notice that even though the “Typical Printer” CDD example above defines 3 capabilities, only 2 are represented in the *Ticket*. That is because the user decided to make a change to only 2 capabilities and not all 3. This is important so that GCP can distinguish capabilities that were actively selected by the user, and others that were left to contain default values.

Here is another example of a ticket produced for the “File Saving Device” example above:

```
{
  “version”: “1.0”,
  “vendor_ticket_item”: [
    {
      “id”: “folder-path”,
      “value”: “~/Documents”
    },
    {
      “id”: “filename”,
      “value”: “mytest.pdf”
    }
  ]
}
```

Notice that the “File Saving Device” does not make use of the *printer* or *scanner* section. That’s because this device is neither of those, and so its capabilities must be expressed in the *vendor_capability* section of the CDD. Consequently, its ticket items are expressed in the *vendor_ticket_item* section of the CJT.

3.5 Ticket Transformation

This is a more advanced topic that most cloud device developers need not worry about. Some cloud applications might pre-process the print job before it reaches the printer. Two examples of such applications are the GCP connector, and the GCP server itself. The GCP server might for example, consume the “page range” ticket item, by stripping pages from a submitted PDF and resetting the “page range” ticket item to {“start”: 1}. Or the GCP connector might generate multiple copies of a document locally (if the printer doesn’t support printing multiple copies). In each of these cases, the ticket is transformed to reflect the change to the document. Potentially, many such applications might transform the ticket before the final document and ticket reach the printer device.

4 Support for Legacy Capabilities Formats

4.1 PPD and XPS Translation Tools

In case your cloud printer uses a capability native format like PPD or XPS, GCP offers web tools that can automatically translate PPD and XPS files into CDD, and can translate CJT back into native ticket formats (JSON for PPD and a psf:PrintTicket document for XPS).

4.2 XPS and CDD Example

Here's a side by side comparison of an XPS capability file taken from a cloud-connected printer and its equivalent CDD:

```
<psf:PrintCapabilities xmlns:psf="..." xmlns:xsi="..." xmlns:xsd="..."
xmlns:ns0000="..." xmlns:psk="..." xmlns:bpe="..." version="1">
  <psf:Feature name="psk:PageMediaType">
    <psf:Property name="ns0000:Manufacturer">
      <psf:Value xsi:type="xsd:string">Canon</psf:Value>
    </psf:Property>
    <psf:Property name="psf:SelectionType">
      <psf:Value xsi:type="xsd:QName">psk:PickOne</psf:Value>
    </psf:Property>
    <psf:Property name="psk:DisplayName">
      <psf:Value xsi:type="xsd:string">Media Type</psf:Value>
    </psf:Property>
    <psf:Option name="psk:Plain" constrained="psk:None">
      <psf:Property name="psk:DisplayName">
        <psf:Value xsi:type="xsd:string">Plain Paper</psf:Value>
      </psf:Property>
      <psf:ScoredProperty name="ns0000:BorderlessPrinting">
        <psf:Value xsi:type="xsd:string">Supported</psf:Value>
      </psf:ScoredProperty>
      <psf:ScoredProperty name="ns0000:CDRPrinting">
        <psf:Value xsi:type="xsd:string">None</psf:Value>
      </psf:ScoredProperty>
    </psf:Option>
    <psf:Option name="ns0000:Glossy" constrained="psk:None">
      <psf:Property name="psk:DisplayName">
        <psf:Value xsi:type="xsd:string">Glossy Photo Paper</
        psf:Value>
      </psf:Property>
      <psf:ScoredProperty name="ns0000:BorderlessPrinting">
        <psf:Value xsi:type="xsd:string">Supported</psf:Value>
      </psf:ScoredProperty>
      <psf:ScoredProperty name="ns0000:CDRPrinting">
        <psf:Value xsi:type="xsd:string">None</psf:Value>
      </psf:ScoredProperty>
    </psf:Option>
  </psf:Feature>
  <psf:Feature name="psk:PageOutputColor">
   <psf:Property name="psf:SelectionType">
     <psf:Value xsi:type="xsd:QName">psk:PickOne</psf:Value>
   </psf:Property>
   <psf:Property name="psk:DisplayName">
     <psf:Value xsi:type="xsd:string">Color</psf:Value>
   </psf:Property>
   <psf:Option name="psk:Color" constrained="psk:None">
     <psf:Property name="psk:DisplayName">
```

```
{
  “version”: “1.0”,
  “vendor_property”: [
    {
      “id”: “ns0000:Manufacturer”,
      “value”: “Canon”
    }
  ],
  “vendor_capability”: [
    {
      “id”: “psk:MediaType”,
      “display_name”: “Media Type”,
      “type”: 1, /*SELECT*/
      “select_cap”: {
        “option”: [
          {
            “value”: “psk:Plain”,
            “display_name”: “Plain Paper”,
            “is_default”: true
          },
          {
            “value”: “ns0000:Glossy”,
            “display_name”: “Glossy Photo”
          }
        ]
      }
    }
  ],
  “printer”: {
    “color”: {
      “option”: [
        {
          “vendor_id”: “psk:Color”,
          “type”: 0, /*STANDARD_COLOR*/
          “is_default”: true
        },
        {
          “vendor_id”: “psk:Monochrome”,
          “type”: 1, /*STANDARD_MONOCHROME*/
        }
      ]
    },
    “duplex”: {
      “option”: [
        {
          “type”: 0, /*NO_DUPLEX*/
          “is_default”: true
        },
        { “type”: 1, /*LONG_EDGE*/ },
        { “type”: 2, /*SHORT_EDGE*/ }
      ]
    },
```

```
      <psf:Value xsi:type="xsd:string">Color</psf:Value>
    </psf:Property>
  </psf:Option>
  <psf:Option name="psk:Monochrome" constrained="psk:None">
    <psf:Property name="psk:DisplayName">
      <psf:Value xsi:type="xsd:string">Grayscale</psf:Value>
    </psf:Property>
  </psf:Option>
 </psf:Feature>
 <psf:Feature name="psk:JobDuplexAllDocumentsContiguously">
  <psf:Property name="psf:SelectionType">
    <psf:Value xsi:type="xsd:QName">psk:PickOne</psf:Value>
  </psf:Property>
  <psf:Property name="psk:DisplayName">
    <psf:Value xsi:type="xsd:string">Duplex Printing</psf:Value>
  </psf:Property>
  <psf:Option name="psk:OneSided" constrained="psk:None">
    <psf:Property name="psk:DisplayName">
      <psf:Value xsi:type="xsd:string">None</psf:Value>
    </psf:Property>
  </psf:Option>
  <psf:Option name="psk:TwoSidedLongEdge" constrained="psk:None">
    <psf:Property name="psk:DisplayName">
      <psf:Value xsi:type="xsd:string">Duplex Printing</psf:Value>
    </psf:Property>
  </psf:Option>

 <psf:Option name="psk:TwoSidedShortEdge"
 constrained="psk:PrintTicketSettings">
    <psf:Property name="psk:DisplayName">
      <psf:Value xsi:type="xsd:string">Duplex Printing</psf:Value>
    </psf:Property>
  </psf:Option>
 </psf:Feature>
 <psf:Feature name="psk:PageOrientation">
  <psf:Property name="psf:SelectionType">
    <psf:Value xsi:type="xsd:QName">psk:PickOne</psf:Value>
  </psf:Property>
  <psf:Property name="psk:DisplayName">
    <psf:Value xsi:type="xsd:string">Orientation</psf:Value>
  </psf:Property>
  <psf:Option name="psk:Portrait" constrained="psk:None">
    <psf:Property name="psk:DisplayName">
      <psf:Value xsi:type="xsd:string">Portrait</psf:Value>
    </psf:Property>
  </psf:Option>
  <psf:Option name="psk:Landscape" constrained="psk:None">
    <psf:Property name="psk:DisplayName">
      <psf:Value xsi:type="xsd:string">Landscape</psf:Value>
    </psf:Property>
  </psf:Option>
 </psf:Feature>
 <psf:ParameterDef name="psk:JobCopiesAllDocuments">
  <psf:Property name="psk:DisplayName">
    <psf:Value xsi:type="xsd:string">Copies</psf:Value>
  </psf:Property>
  <psf:Property name="psf:DataType">
    <psf:Value xsi:type="xsd:QName">xsd:integer</psf:Value>
  </psf:Property>
  <psf:Property name="psf:Multiple">
    <psf:Value xsi:type="xsd:integer">1</psf:Value>
  </psf:Property>
  <psf:Property name="psf:MaxValue">
    <psf:Value xsi:type="xsd:integer">999</psf:Value>
  </psf:Property>
```

```
    “page_orientation”: {
      “option”: [
        {
          “type”: 0 /*PORTRAIT*/,
          “is_default”: true
        },
        { “type”: 1 /*LANDSCAPE*/ }
      ]
    },
    “copies”: {
      “default”: 1,
      “max”: 999
    },
    “dpi”: {
      “option”: [
        {
          “type”: 2, /*NORMAL*/
          “horizontal_dpi”: 300,
          “vertical_dpi”: 300,
          “is_default”: true
        },
        {
          “type”: 4, /*BEST*/
          “horizontal_dpi”: 600,
          “vertical_dpi”: 600
        }
      ]
    },
    “fit_to_page”: {
      “option”: [
        {
          “type”: 0, /*NO_FITTING*/
          “is_default”: true
        },
        { “type”: 1, /*FIT_TO_PAGE*/ }
      ]
    },
    “media_size”: {
      “option”: [
        {
          “type”: 8, /*LETTER*/
          “width_microns”: 215900,
          “height_microns”: 279400,
          “is_default”: true
        },
        {
          “type”: 9, /*LEGAL*/
          “width_microns”: 215900,
          “height_microns”: 355600
        },
        {
          “type”: 7, /*A5*/
          “width_microns”: 148000,
          “height_microns”: 210000
        },
        {
          “type”: 6, /*A4*/
          “width_microns”: 210000,
          “height_microns”: 297000
        },
        {
          “type”: 0, /*CUSTOM*/
          “width_microns”: 55000,
          “height_microns”: 91000
        },
        {
          “type”: 4, /*A3*/
          “width_microns”: 297000,
          “height_microns”: 420000
        },
        {
          “type”: 3, /*A2*/
          “width_microns”: 420000,
          “height_microns”: 594000
        }
```

```
  <psf:Property name="psf:MinValue">
    <psf:Value xsi:type="xsd:integer">1</psf:Value>
  </psf:Property>
  <psf:Property name="psf:DefaultValue">
    <psf:Value xsi:type="xsd:integer">1</psf:Value>
  </psf:Property>
  <psf:Property name="psf:Mandatory">
    <psf:Value xsi:type="xsd:QName">psk:Unconditional</psf:Value>
  </psf:Property>
  <psf:Property name="psf:UnitType">
    <psf:Value xsi:type="xsd:string">copies</psf:Value>
  </psf:Property>
  <psf:Property name="ns0000:State">
    <psf:Value xsi:type="xsd:string">Enable</psf:Value>
  </psf:Property>
 </psf:ParameterDef>
 <psf:Feature name="psk:PageResolution">
  <psf:Property name="psf:SelectionType">
    <psf:Value xsi:type="xsd:QName">psk:PickOne</psf:Value>
  </psf:Property>
  <psf:Property name="psk:DisplayName">
    <psf:Value xsi:type="xsd:string">Resolution</psf:Value>
  </psf:Property>
  <psf:Option name="ns0000:r600x600" constrained="psk:None">
    <psf:Property name="psk:DisplayName">
      <psf:Value xsi:type="xsd:string">600x600</psf:Value>
    </psf:Property>
    <psf:ScoredProperty name="psk:ResolutionX">
      <psf:Value xsi:type="xsd:integer">600</psf:Value>
    </psf:ScoredProperty>
    <psf:ScoredProperty name="psk:ResolutionY">
      <psf:Value xsi:type="xsd:integer">600</psf:Value>
    </psf:ScoredProperty>
  </psf:Option>
  <psf:Option name="ns0000:r300x300" constrained="psk:None">
    <psf:Property name="psk:DisplayName">
      <psf:Value xsi:type="xsd:string">300x300</psf:Value>
    </psf:Property>
    <psf:ScoredProperty name="psk:ResolutionX">
      <psf:Value xsi:type="xsd:integer">300</psf:Value>
    </psf:ScoredProperty>
    <psf:ScoredProperty name="psk:ResolutionY">
      <psf:Value xsi:type="xsd:integer">300</psf:Value>
    </psf:ScoredProperty>
  </psf:Option>
 </psf:Feature>
 <psf:Feature name="psk:PageScaling">
  <psf:Property name="psf:SelectionType">
    <psf:Value xsi:type="xsd:QName">psk:PickOne</psf:Value>
  </psf:Property>
  <psf:Property name="psk:DisplayName">
    <psf:Value xsi:type="xsd:string">Scaled</psf:Value>
  </psf:Property>
  <psf:Option name="psk:None" constrained="psk:None">
    <psf:Property name="psk:DisplayName">
      <psf:Value xsi:type="xsd:string">Off</psf:Value>
    </psf:Property>
  </psf:Option>
  <psf:Option name="ns0000:Fitpage" constrained="psk:None">
    <psf:Property name="psk:DisplayName">
      <psf:Value xsi:type="xsd:string">Fit-to-Page</psf:Value>
    </psf:Property>
  </psf:Option>
 </psf:Feature>
 <psf:Feature name="psk:PageMediaSize">
```

```
      ]
    },
    “collate”: { “default”: false },
    “reverse_order”: { “default”: false }
  }
}
```

```
 <psf:Property name="psf:SelectionType">
   <psf:Value xsi:type="xsd:QName">psk:PickOne</psf:Value>
 </psf:Property>
 <psf:Property name="psk:DisplayName">
   <psf:Value xsi:type="xsd:string">Page Size</psf:Value>
 </psf:Property>
 <psf:Option name="psk:NorthAmericaLetter" constrained="psk:None">
   <psf:Property name="psk:DisplayName">
     <psf:Value xsi:type="xsd:string">Letter 8.5"x11"</psf:Value>
   </psf:Property>
   <psf:ScoredProperty name="psk:MediaSizeWidth">
     <psf:Value xsi:type="xsd:integer">215900</psf:Value>
   </psf:ScoredProperty>
   <psf:ScoredProperty name="psk:MediaSizeHeight">
     <psf:Value xsi:type="xsd:integer">279400</psf:Value>
   </psf:ScoredProperty>
 </psf:Option>
 <psf:Option name="psk:NorthAmericaLegal" constrained="psk:None">
   <psf:Property name="psk:DisplayName">
     <psf:Value xsi:type="xsd:string">Legal</psf:Value>
   </psf:Property>
   <psf:ScoredProperty name="psk:MediaSizeWidth">
     <psf:Value xsi:type="xsd:integer">215900</psf:Value>
   </psf:ScoredProperty>
   <psf:ScoredProperty name="psk:MediaSizeHeight">
     <psf:Value xsi:type="xsd:integer">355600</psf:Value>
   </psf:ScoredProperty>
 </psf:Option>
 <psf:Option name="psk:ISOA5" constrained="psk:None">
   <psf:Property name="psk:DisplayName">
     <psf:Value xsi:type="xsd:string">A5</psf:Value>
   </psf:Property>
   <psf:ScoredProperty name="psk:MediaSizeWidth">
     <psf:Value xsi:type="xsd:integer">148000</psf:Value>
   </psf:ScoredProperty>
   <psf:ScoredProperty name="psk:MediaSizeHeight">
     <psf:Value xsi:type="xsd:integer">210000</psf:Value>
   </psf:ScoredProperty>
 </psf:Option>
 <psf:Option name="psk:ISOA4" constrained="psk:None">
   <psf:Property name="psk:DisplayName">
     <psf:Value xsi:type="xsd:string">A4</psf:Value>
   </psf:Property>
   <psf:ScoredProperty name="psk:MediaSizeWidth">
     <psf:Value xsi:type="xsd:integer">210000</psf:Value>
   </psf:ScoredProperty>
   <psf:ScoredProperty name="psk:MediaSizeHeight">
     <psf:Value xsi:type="xsd:integer">297000</psf:Value>
   </psf:ScoredProperty>
 </psf:Option>
 <psf:Option name="psk:BusinessCard" constrained="psk:None">
   <psf:Property name="psk:DisplayName">
     <psf:Value xsi:type="xsd:string">Card 2.16"x3.58" 55x91mm</
psf:Value>
   </psf:Property>
   <psf:ScoredProperty name="psk:MediaSizeWidth">
     <psf:Value xsi:type="xsd:integer">55000</psf:Value>
   </psf:ScoredProperty>
   <psf:ScoredProperty name="psk:MediaSizeHeight">
     <psf:Value xsi:type="xsd:integer">91000</psf:Value>
   </psf:ScoredProperty>
 </psf:Option>

<psf:Option name="ns0000:A4Plus"
constrained="psk:PrintTicketSettings">
```

```
    <psf:Property name="psk:DisplayName">
      <psf:Value xsi:type="xsd:string">A4+ (Scaled)</psf:Value>
    </psf:Property>
    <psf:ScoredProperty name="psk:MediaSizeWidth">
      <psf:Value xsi:type="xsd:integer">222700</psf:Value>
    </psf:ScoredProperty>
    <psf:ScoredProperty name="psk:MediaSizeHeight">
      <psf:Value xsi:type="xsd:integer">355600</psf:Value>
    </psf:ScoredProperty>
  </psf:Option>
  <psf:Option name="psk:ISOA3" constrained="psk:PrintTicketSettings">
    <psf:Property name="psk:DisplayName">
      <psf:Value xsi:type="xsd:string">A3 (Scaled)</psf:Value>
    </psf:Property>
    <psf:ScoredProperty name="psk:MediaSizeWidth">
      <psf:Value xsi:type="xsd:integer">297000</psf:Value>
    </psf:ScoredProperty>
    <psf:ScoredProperty name="psk:MediaSizeHeight">
      <psf:Value xsi:type="xsd:integer">420000</psf:Value>
    </psf:ScoredProperty>
  </psf:Option>
  <psf:Option name="psk:ISOA2" constrained="psk:PrintTicketSettings">
    <psf:Property name="psk:DisplayName">
      <psf:Value xsi:type="xsd:string">A2 (Scaled)</psf:Value>
    </psf:Property>
    <psf:ScoredProperty name="psk:MediaSizeWidth">
      <psf:Value xsi:type="xsd:integer">420000</psf:Value>
    </psf:ScoredProperty>
    <psf:ScoredProperty name="psk:MediaSizeHeight">
      <psf:Value xsi:type="xsd:integer">594000</psf:Value>
    </psf:ScoredProperty>
  </psf:Option>
  <psf:Option name="psk:CustomMediaSize" constrained="psk:None">
    <psf:Property name="psk:DisplayName">
      <psf:Value xsi:type="xsd:string">Custom...</psf:Value>
    </psf:Property>
    <psf:ScoredProperty name="psk:MediaSizeWidth">
      <psf:ParameterRef name="psk:PageMediaSizeMediaSizeWidth"/>
    </psf:ScoredProperty>
    <psf:ScoredProperty name="psk:MediaSizeHeight">
      <psf:ParameterRef name="psk:PageMediaSizeMediaSizeHeight"/>
    </psf:ScoredProperty>
  </psf:Option>
 </psf:Feature>
 <psf:Feature name="psk:DocumentCollate">
  <psf:Property name="psf:SelectionType">
    <psf:Value xsi:type="xsd:QName">psk:PickOne</psf:Value>
  </psf:Property>
  <psf:Property name="psk:DisplayName">
    <psf:Value xsi:type="xsd:string">Collate</psf:Value>
  </psf:Property>
  <psf:Option name="psk:Uncollated" constrained="psk:None">
    <psf:Property name="psk:DisplayName">
      <psf:Value xsi:type="xsd:string">Off</psf:Value>
    </psf:Property>
  </psf:Option>
  <psf:Option name="psk:Collated" constrained="psk:None">
    <psf:Property name="psk:DisplayName">
      <psf:Value xsi:type="xsd:string">On</psf:Value>
    </psf:Property>
  </psf:Option>
 </psf:Feature>
 <psf:Feature name="psk:JobPageOrder">
  <psf:Property name="psf:SelectionType">
    <psf:Value xsi:type="xsd:QName">psk:PickOne</psf:Value>
```

```
   </psf:Property>
   <psf:Property name="psk:DisplayName">
     <psf:Value xsi:type="xsd:string">Print from Last Page</psf:Value>
   </psf:Property>
   <psf:Option name="psk:Standard" constrained="psk:None">
     <psf:Property name="psk:DisplayName">
       <psf:Value xsi:type="xsd:string">Off</psf:Value>
     </psf:Property>
   </psf:Option>
   <psf:Option name="psk:Reverse" constrained="psk:None">
     <psf:Property name="psk:DisplayName">
       <psf:Value xsi:type="xsd:string">On</psf:Value>
     </psf:Property>
   </psf:Option>
  </psf:Feature>
</psf:PrintCapabilities>
```

5 Appendix

5.1 Printer Section Objects

```
// Property that defines what content types the printer can print natively.
message SupportedContentType {
  message Option {
    // Content type (e.g. "image/png" or "application/pdf"). Use “*/*”, if your
    // printer supports all formats.
    required string content_type = 1;

    // Numeric rank used to order content types. Content types with lower rank
    // rank will be used before those with higher rank. Minimum rank should be
    // 1.
    required int32 rank = 2;

    // Minimum supported version of the content type if applicable (e.g. "1.5").
    optional string min_version = 3;

    // Maximum supported version of the content type if applicable (e.g. "1.5").
    optional string max_version = 3;
  }

  repeated Option option = 1;
}

// Property that defines what speeds (in pages per minute) the printer can
// operate at.
message PrintingSpeed {
  // Available speed of the printer.
  //
  // Specify settings that are associated with the given speed. If a setting
  // is left unset, then it will be assumed that the speed is independent of
  // that setting. For example, the following Option
  //
  //   {
  //     “speed_ppm”: 5.5,
  //     “color_type”: [1 /*STANDARD_MONOCHROME*/],
  //     “dpi_type”: [],
  //     “media_size_type”: [8 /*LETTER*/, 6 /*A4*/]
  //   }
  //
  // indicates that the printer prints at 5.5 pages per minute when printing in
  // MONOCHROME in either LETTER or A4 paper sizes, but does not depend on any
  // particular print quality.
  message Option {
    // Speed measured in pages per minute.
    required float speed_ppm = 1;

    // Types of color settings that operates at this speed.
    repeated Color.Type color_type = 2;

    // Types of print quality settings that operates at this speed.
    repeated Dpi.Type dpi_type = 3;

    // Types of color settings that operates at this speed.
```

```
    repeated MediaSize.Type media_size_type = 4;
  }

  // Speeds that the printer can operate at.
  repeated Option option = 1;
}

// Configuration of how printer should receive PWG raster images.
message PwgRasterConfig {
  // Type of flip to perform on a page.
  enum FlipType {
    NONE = 0;
    LONG_EDGE = 1;
    SHORT_EDGE = 2;
  }

  // Whether this printer accepts all of its pages rotated by 180 degrees.
  // Ex. 1' 2' 3' 4' where ' means rotated.
  optional bool rotate_all_pages_180 = 1 [default = false];

  // Whether this printer accepts even pages rotated by 180 degrees when
  // printing in duplex. Ex. 1 2' 3 4' where ' means rotated.
  optional bool rotate_even_pages_180_for_duplex = 2 [default = false];

  // Whether this printer accepts even pages flipped when printing in duplex.
  // Ex. 1 2^ 3 4^ where ^ means flipped.
  optional FlipType flip_even_pages_for_duplex = 3 [default = NONE];

  // Whether this printer accepts printing even pages before odd pages when
  // printing in duplex. Ex. 2 1 4 3.
  optional bool print_even_page_first_for_duplex = 4 [default = false];

  // Whether this printer needs pages flipped before being rotated when
  // printing in duplex. This only applies if both flip_even_pages_for_duplex
  // and rotate_even_pages_180_for_duplex are both true.
  optional bool flip_first_then_rotate_for_duplex = 5 [default = false];
}

// Capability that defines a set of duplexing options available on a device.
message Duplex {
  enum Type {
    NO_DUPLEX = 0;
    LONG_EDGE = 1;
    SHORT_EDGE = 2;
    MANUAL_DUPLEX = 3;
  }

  message Option {
    required Type type = 1;
    optional bool is_default = 2 [default = false];
  }

  repeated Option option = 1;
}

// Capability that defines a set of page-orientations options available on a
// device.
message PageOrientation {
  enum Type {
    PORTRAIT = 0;
```

```
    LANDSCAPE = 1;
    REVERSE_PORTRAIT = 2;
    REVERSE_LANDSCAPE = 3;
  }

  message Option {
    required Type type = 1;
    optional bool is_default = 2 [default = false];
  }

  repeated Option option = 1;
}

// Capability that defines a default and maximum value for multiple copies on a
// device.
message Copies {
  optional int32 default = 1;
  optional int32 max = 2;
}

// Capability that defines a set of margins available on a device (including a
// custom one). Margins are measured in microns.
message Margins {
  enum Type {
    BORDERLESS = 0;
    STANDARD = 1;
    CUSTOM = 2;
  }

  message Option {
    required Type type = 1;
    required int32 top_microns = 2;
    required int32 right_microns = 3;
    required int32 bottom_microns = 4;
    required int32 left_microns = 5;
    optional bool is_default = 6 [default = false];
  }

  repeated Option option = 1;
  optional int32 min_top_microns = 2;
  optional int32 min_right_microns = 3;
  optional int32 min_bottom_microns = 4;
  optional int32 min_left_microns = 5;
}

// Capability that defines a set of page fitting options available on a device.
message FitToPage {
  enum Type {
    NO_FITTING = 0;
    FIT_TO_PAGE = 1;
    GROW_TO_PAGE = 2;
    SHRINK_TO_PAGE = 3;
  }

  message Option {
    required Type type = 1;
    optional bool is_default = 2 [default = false];
  }

  repeated Option option = 1;
```

```
}

// Whether to rotate pages to best fit a media size. This is useful for
// documents that have pages in different orientations.
message RotateToPage {
  // Whether to automatically rotate pages by default.
  required bool default = 1;
}

// Capability that defines a default page-range selection on a device.
message PageRange {

  // Interval of pages in the document to print.
  message Interval {
    // Beginning of the interval (inclusive).
    required int32 start = 1;

    // End of the interval (inclusive). If not set, then the interval will
    // include all available pages after start.
    optional int32 end = 2;
  }

  repeated Interval default = 1;
}

// Capability that defines the default collation setting on a device.
message Collate {
  required bool default = 1;
}

// Capability that defines the default reverse-printing-order setting on a device.
message ReverseOrder {
  required bool default = 1;
}

// Section of a CJT of how a print job should be handled by a cloud-connected
// printer.
message PrinterTicketSection {
  optional ColorTicketItem color = 1;
  optional DuplexTicketItem duplex = 2;
  optional PageOrientationTicketItem page_orientation = 3;
  optional CopiesTicketItem copies = 4;
  optional MarginsTicketItem margins = 5;
  optional DpiTicketItem dpi = 6;
  optional FitToPageTicketItem fit_to_page = 7;
  optional PageRangeTicketItem page_range = 8;
  optional MediaSizeTicketItem media_size = 9;
  optional CollateTicketItem collate = 10;
  optional ReverseOrderTicketItem reverse_order = 11;
  optional RotateToPageTicketItem rotate_to_page = 12;
}

// Ticket item indicating which duplexing option to use.
message DuplexTicketItem {
  required Duplex.Type type = 1;
}

// Ticket item indicating which page orientation option to use.
message PageOrientationTicketItem {
  required PageOrientation.Type type = 1;
```

```
}

// Ticket item indicating how many copies to produce.
message CopiesTicketItem {
  required int32 copies = 1;
}

// Ticket item indicating what margins to use (in microns).
message MarginsTicketItem {
  required int32 top_microns = 1;
  required int32 right_microns = 2;
  required int32 bottom_microns = 3;
  required int32 left_microns = 4;
}

// Ticket item indicating what page-fitting algorithm to use.
message FitToPageTicketItem {
  required FitToPage.Type type = 1;
}

// Ticket item indicating whether to automatically rotate pages.
message RotateToPageTicketItem {
  required bool rotate_to_page = 1;
}

// Ticket item indicating what pages to use.
message PageRangeTicketItem {
  repeated PageRange.Interval interval = 1;
}

// Ticket item indicating whether to collate pages.
message CollateTicketItem {
  required bool collate = 1;
}

// Ticket item indicating whether to print in reverse.
message ReverseOrderTicketItem {
  required bool reverse_order = 1;
}
```

5.2 Scanner Section Objects

```
// Section of a CDD that describes the capabilities of a cloud-connected
// scanner.
message ScannerDescriptionSection {

  // Color scanning capabilities of the scanner.
  optional Color color = 1;

  // Image quality or dots-per-inch capabilities of the scanner.
  optional Dpi dpi = 2;

  // Image size capabilities of the scanner.
  optional MediaSize media_size = 3;

  // File format type capabilities of the scanner.
  optional FileFormat file_format = 4;
```

```
}

// Capability that defines a set file format available on a scanner.
message FileFormat {
  enum Type {
    CUSTOM = 0;
    JPEG = 1;
    PDF = 2;
    PNG = 3;
    TIFF = 4;
  }

  message Option {
    required Type type = 1;

    // Content type (or MIME type) of the option. Should only be used with the
    // Type.CUSTOM type.
    optional string custom_content_type = 2;

    optional bool is_default = 3 [default = false];
  }

  repeated Option option = 1;
}

// Section of a CJT of how a scan job should be handled by the cloud-connected
// scanner.
message ScannerTicketSection {
  optional ColorTicketItem color = 1;
  optional DpiTicketItem dpi = 2;
  optional MediaSizeTicketItem media_size = 3;
  optional FileTypeTicketItem file_type = 4;
}

// Ticket item indicating what pages to use.
message FileTypeTicketItem {
  required FileFormat.Type type = 1;
}
```

5.3 Common Objects

```
// Capability that defines a set of color options available on a device.
message Color {
  enum Type {
    STANDARD_COLOR = 0;
    STANDARD_MONOCHROME = 1;
    CUSTOM_COLOR = 2;
    CUSTOM_MONOCHROME = 3;
  }

  message Option {
    // ID to help vendor identify the color option.
    required string vendor_id = 1;

    // Type of color option used in UI to differentiate color and non-color
    // options. Note there should only be at most one STANDARD_COLOR option, at
    // most one STANDARD_MONOCHROME, and any number of the CUSTOM_* options.
    required Type type = 2;
```

```
    // User-friendly string that represents this option. Options marked as
    // STANDARD_COLOR or STANDARD_MONOCHROME will have their display-name
    // localized, this field should be used for CUSTOM_* options.
    optional string custom_display_name = 3;

    // Whether this option should be selected by default. Only one option
    // should be set as default.
    optional bool is_default = 4 [default = false];
  }

  repeated Option option = 1;
}

// Capability that defines a set of 2D image quality levels available on a
// device.
message Dpi {
  enum Type {
    CUSTOM = 0;
    DRAFT = 1;
    NORMAL = 2;
    PHOTO = 3;
    BEST = 4;
  }

  message Option {
    required Type type = 1;
    required int32 horizontal_dpi = 2;
    required int32 vertical_dpi = 3;
    optional bool is_default = 4 [default = false];
  }

  repeated Option option = 1;
  optional int32 min_horizontal_dpi = 2;
  optional int32 max_horizontal_dpi = 3;
  optional int32 min_vertical_dpi = 4;
  optional int32 max_vertical_dpi = 5;
}

// Capability that defines a set of media sizes available on a device.
message MediaSize {
  enum Type {
    CUSTOM = 0;
    A0 = 1;
    A1 = 2;
    A2 = 3;
    A3 = 4;
    A3_PLUS = 5;
    A4 = 6;
    A5 = 7;
    LETTER = 8;
    LEGAL = 9;
    LEDGER = 10;
  }

  message Option {
    required Type type = 1;
    optional int32 width_microns = 2;
    optional int32 height_microns = 3;
    optional bool is_continuous_feed = 4;
    optional bool is_default = 5 [default = false];
```

```
  }

  repeated Option option = 1;
  optional int32 max_width_microns = 2;
  optional int32 max_height_microns = 3;
}

// Ticket item indicating which color option to use.
message ColorTicketItem {
  required string vendor_id = 1;
  required Color.Type type = 2;
}

// Ticket item indicating what image resolution to use.
message DpiTicketItem {
  required int32 horizontal_dpi = 1;
  required int32 vertical_dpi = 2;
}

// Ticket item indicating what media size to use.
message TicketItem {
  required int32 width_microns = 1;
  required int32 height_microns = 2;
}
```

5.4 Integration with GCP

Even though the data structures used to communicate CDDs and CJTs are defined in Protobuf format, HTTP requests with GCP are done over a JSON format that mirrors the Protobuf format. The various examples used in describing the CDD and CJT formats have been given in this JSON format.

There a few GCP APIs where the CDD format can be used:

- Registering devices (/register)
- Getting a device's capabilities (/printer)
- Submitting a print job (/submit)
- Fetching a ticket (/ticket)

5.4.1 Printer Registration

Historically, registering a printer requires a PPD or XPS capability file to be sent along side other printer metadata to register a printer with GCP. GCP now supports receiving a JSON representation of a CDD. The "Typical Printer" example above defines a CDD file that can be used in a /register request in lieu of a PPD or XPS capability file.

5.4.2 Getting Device Capabilities

In order to be backwards compatible, using the /printer API currently returns a JSON representation of the device's capabilities in a legacy format that predates CDD. To receive a device's capabilities in the CDD format, add the "use_cdd_format=true" parameter to your /printer HTTP request. This also works for the /search API (recent printers in the /search request are returned with capabilities).

5.4.3 Submitting a Print Job

Historically, a job submission client application would use the “capabilities” parameter of the /submit API to specify the print ticket. The ticket can now instead be specified in the CJT format using the “ticket” parameter. See section Ticket for an example of a CJT in JSON form.

5.4.4 Getting a Ticket

Historically, cloud devices downloaded their ticket from a “ticketUrl” field specified in the print job object. This legacy ticket was either given in JSON format for PPD printers or XML format for XPS printers. However, no longer do devices need to understand either of these formats since now they just need to understand the CJT format. To get a CJT for a given print job, use the /ticket?jobid=<jobid>&format=CJT API. Here’s an example of the response of a /ticket?jobid=1234&format=CJT request (formatted and commented for convenience):

```
{
  “version”: “1.0”,
  “vendor_ticket_item”: [],
  “printer”: {
    “color”: {
      “vendor_id”: “grayscale”,
      “type”: 1 /*MONOCHROME*/
    },
    “copies”: { “copies”: 3 }
  }
}
```