Cloud Device Description

(Cloud 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 its hierarchy structure. XPS's XML format is verbose and not ideal for internet transfer and use 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 either of the formats. Every time 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 are eventually going to build this in to prevent user from choosing the 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 printing” (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 {

  // Version of the CDD in the form "X.Y" where changes to Y are backwards
  // compatible, and changes to X are not (required).
  optional 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;

  // Section of the CDD that specifically describes printers.
  optional PrinterDescriptionSection printer = 101;

  // Section of the CDD that specifically describes scanners.
  optional ScannerDescriptionSection scanner = 102;
}
```

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 subsections. Initially, these subsections are *printer* and *scanner*. It can be surmised that the CDD can represent a printer or scanner or even both: a printer-scanner combo.

2.1 Device-specific Description Sections

CDD is broken into device-specific subsections that each describe a separate sub-device of the cloud connected device. There is a common structure to how each subsection is laid out. Each subsection is divided into two parts. The first part (field numbers 1 - 100) consist of a read-only description of the sub-device. For example, a read-only description of a printer might be a list of printing speeds that it can operate at, or how it needs PWG rasters sent to it. These descriptions are read-only in the sense that a user of the device cannot change them per cloud job. The second part of the device-specific section consists of the capabilities that the user can change (field numbers 101-200). These fields represent user selectable capabilities or features that are represented in the printing (or scanning) UI. Also, the results of the user selection are sent to the device in the form of a job ticket. Notice the device-specific sections of the job ticket only consist of fields 101-200 and

not any from 1-100.

Another important note about the second part of the subdevice section (fields 101-200). These capability fields are all optional since not all printers might be able to support all of these capabilities. If a printer does not support a particular capability, this field should be left unset. If it is available, then its field should be set. For example, if a printer supports the page range capability, then the *page_range* field of the *PrinterDescriptionSection* must be set. Whether or not the field is set tells the server that the device supports that capability.

2.2 Printer Description Section

This section of 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 and physical units of a
// cloud-connected printer.
message PrinterDescriptionSection {

  // Content types (sometimes referred to as MIME types) that are supported by
  // the printer.
  //
  // The order of these types determines which content type the document should
  // be converted to. For example, if the types are ordered as:
  //
  //   [
  //     {"content_type": "application/pdf"},
  //     {"content_type": "image/pwg-raster"}
  //   ]
  //
  // Then the document's content type will first be matched to any content type
  // in the list. If there is a match, then the document will be sent to the
  // printer as is. If there is no match, then the document will be converted to
  // a content type which the server supports starting from the first option. In
  // this example, if the document is sent as "text/html" and the printer
  // supports "application/pdf" and "image/pwg-raster", then the document will
  // be converted to "application/pdf" and not "image/pwg-raster", because
  // "application/pdf" is declared earlier in this list.
  repeated 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;

  // Physical model of the printer's input trays.
  repeated InputTrayUnit input_tray_unit = 4;

  // Physical model of the printer's output bins.
  repeated OutputBinUnit output_bin_unit = 5;

  // Physical model of the printer's markers.
  repeated Marker marker = 6;
```

```
  // Physical model of the printer's covers.
  repeated Cover cover = 7;

  // Physical model of the printer's media paths.
  repeated MediaPath media_path = 8;

  // Vendor-provided printer capabilities.
  repeated VendorCapability vendor_capability = 101;

  // Color printing capabilities of the printer.
  optional Color color = 102;

  // Duplexing capabilities of the printer.
  optional Duplex duplex = 103;

  // Page/paper orientation capabilities of the printer.
  optional PageOrientation page_orientation = 104;

  // Multiple copy capability of the printer.
  optional Copies copies = 105;

  // Page margins capability of the printer.
  optional Margins margins = 106;

  // Printing quality or dots-per-inch (DPI) capabilities of the printer.
  optional Dpi dpi = 107;

  // Page fitting capabilities of the printer.
  optional FitToPage fit_to_page = 108;

  // Page range selection capability of the printer.
  optional PageRange page_range = 109;

  // Page or media size capabilities of the printer.
  optional MediaSize media_size = 110;

  // Paper collation capability of the printer.
  optional Collate collate = 111;

  // Reverse order printing capability of the printer.
  optional ReverseOrder reverse_order = 112;
}
```

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 set one of the fields above.

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

The *vendor_capability* section represents all other capabilities that are currently not semantically understood in CDD. *vendor_capability* is a free-form section similar to PPD or XPS where new capabilities can be represented.

The printer-specific capability proto definitions are given in detail in the Printer Section Objects section. For example, here is a proto definition 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;
  }

  message Option {
    optional Type type = 1 [default = NO_DUPLEX];
    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.

2.3 Scanner Description Section

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 {

  // Vendor-provided printer capabilities.
  repeated VendorCapability vendor_capability = 101;

  // Color scanning capabilities of the scanner.
  optional Color color = 102;

  // Image quality or dots-per-inch capabilities of the scanner.
  optional Dpi dpi = 103;

  // Image size capabilities of the scanner.
  optional MediaSize media_size = 104;

  // File format type capabilities of the scanner.
  optional FileFormat file_format = 105;
}
```

New capabilities common to most scanners will be added to this protobuf. These existing capabilities are described in detail in the Scanner Section Objects. 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 {
  message Option {
    // Horizontal DPI (required).
    optional int32 horizontal_dpi = 1;

    // Vertical DPI (required).
    optional int32 vertical_dpi = 2;

    optional bool is_default = 3 [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;
}
```

For example, 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.

2.4 Vendor Capabilities

If a cloud-connected device exhibits capabilities that don't belong into any of the semantic sections of 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. Vendors can store capabilities for a printer or scanner in this section that are not yet semantically supported in CDD. For example: media type or printing multiple pages per sheet. Here's the protobuf definition of a vendor capability:

```
// 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).
  optional string id = 1;

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

  // Type of this capability (required).
  optional 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 are represented using 3 types of values. The option which is used depends on how the vendor-specific capability is presented in the UI. Currently, three are supported:

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

In the UI a range capability might look like a slider control. A typed-value capability might just be a text box that the user can type in. And a selection based capability would look like a drop-down list of options. The *type* field should be set with the appropriate *VendorCapability.Type* enum value and the corresponding *_cap field should be initialized with the appropriate data.

2.4.1 Range Based

A range-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;
  }

  // Data type of the value of the range capability (required).
  optional ValueType value_type = 1;
  optional string default = 2;
  optional string min = 3;
  optional string max = 4;
}
```

2.4.2 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;
```

```
  }

  // Type of data of the typed value capability (required).
  optional ValueType value_type = 1;
  optional string default = 2;
}
```

2.4.3 Selection Based

One of the most common capabilities (of printers) are selection based capabilities. These capabilities present a list of predefined options for a 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).
    optional string value = 1;

    // User-friendly string to represent this option (required).
    optional 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;
}
```

2.5 Examples

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

2.5.1 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",
```

```
  "printer": {
    "supported_content_type": [
        {"content_type": "application/pdf", "min_version": "1.5"},
        {"content_type": "image/jpeg"},
        {"content_type": "text/plain"}
    ],
    "vendor_capability": [],
    "color": {
      "option": [
        {"type": "STANDARD_MONOCHROME"},
        {"type": "STANDARD_COLOR", "is_default": true},
        {
          "vendor_id": "ultra-color",
          "type": "CUSTOM_COLOR",
          "custom_display_name": "Best Color"
        },
      ]
    },
    "copies": {
      "default": 1,
      "max": 511
    },
    "media_size": {
      "option": [
        {
          "name": "ISO_A4",
          "width_microns": 210000,
          "height_microns": 297000,
          "is_default": true
        },
        {
          "name": "NA_LEGAL",
          "width_microns": 215900,
          "height_microns": 355600
        },
        {
          "name": "NA_LETTER",
          "width_microns": 215900,
          "height_microns": 279400
        }
      ],
    }
  ],
}
```

2.5.2 File Saving Device

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

```
{
  "version": "1.0",
  "printer": {
    "supported_content_type": [ { "content_type": "application/pdf" } ],
    "vendor_capability": [
      {
```

```
        "id": "folder-path",
        "display_name": "Destination Folder",
        "type": "TYPED_VALUE",
        "typed_value_cap": {
          "value_type": "STRING",
          "default": "/tmp/"
        }
      },
      {
        "id": "filename",
        "display_name": "File Name",
        "type": "TYPED_VALUE",
        "typed_value_cap": {
          "value_type": "STRING",
          "default": "printout.pdf"
        }
      }
    ],
  }
}
```

2.6 CDD Visualization Tool

GCP offers a web-based tool that helps CDD authors visualize what our UI will look like given a valid CDD. This tool is available at https://www.google.com/cloudprint/tools/cdd/cdd.html.

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

  // Section of CJT pertaining to cloud printer ticket items.
  optional PrinterTicketSection print = 101;

  // Section of CJT pertaining to cloud scanner ticket items.
  optional ScannerTicketSection scan = 102;
}
```

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 {
  repeated VendorTicketItem vendor_ticket_item = 1;
  optional ColorTicketItem color = 2;
  optional DuplexTicketItem duplex = 3;
  optional PageOrientationTicketItem page_orientation = 4;
  optional CopiesTicketItem copies = 5;
  optional MarginsTicketItem margins = 6;
  optional DpiTicketItem dpi = 7;
  optional FitToPageTicketItem fit_to_page = 8;
  optional PageRangeTicketItem page_range = 9;
  optional MediaSizeTicketItem media_size = 10;
  optional CollateTicketItem collate = 11;
  optional ReverseOrderTicketItem reverse_order = 12;
}
```

The protobuf definition of each ticket item can be found in the Printer Section Objects section. 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 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 therefore don't belong in the CJT. The *margins* ticket item is one of the more complicated ones. Its protobuf definition is reproduced here as an example:

```
// Ticket item indicating what margins to use (in microns).
message MarginsTicketItem {
  // Top margin of the page (required).
  optional int32 top_microns = 1;

  // Top margin of the page (required).
  optional int32 right_microns = 2;

  // Top margin of the page (required).
  optional int32 bottom_microns = 3;

  // Top margin of the page (required).
  optional int32 left_microns = 4;
}
```

Let's explore the *vendor* repeated field. This repeated field contains all of the vendor-specific capability selections from the user and correspond to the capabilities supplied in the *PrinterDescriptionSection.vendor_capability* repeated field.

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 {
  repeated VendorTicketItem vendor_ticket_item = 1;
  optional ColorTicketItem color = 2;
  optional DpiTicketItem dpi = 3;
  optional MediaSizeTicketItem media_size = 4;
  optional FileTypeTicketItem file_type = 5;
}
```

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 CDD, the CJT device-specific sections contain 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).
  optional string id = 1;

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

3.4 Examples

Here is a sample ticket produced for the “Typical Printer” example above:

```
{
  "version": "1.0",
  "print": {
    "vendor_ticket_item": [],
    "color": {"type": "STANDARD_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 sample ticket produced for the “File Saving Device” example above:

```
{
  "version": "1.0",
  "print": {
    "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* section’s semantic fields. Because this device is only using vendor-specific capabilities, these capabilities must be expressed in the *vendor_capability* section of CDD. Consequently, its ticket items are expressed in the *vendor_ticket_item* section of the CJT *printer* section.

3.5 Ticket Transformation

This is an advanced topic that cloud device developers will seldom need knowledge of. 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). These tools are located at https://www.google.com/cloudprint/tools/cdd/cdd.html.

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">
```

```
{
  "version": "1.0",
  "printer": {
    "vendor_capability": [
      {
        "id": "psk:MediaType",
        "display_name": "Media Type",
        "type": "SELECT",
        "select_cap": {
          "option": [
            {
              "value": "psk:Plain",
              "display_name": "Plain Paper",
              "is_default": true
            },
            {
              "value": "ns0000:Glossy",
              "display_name": "Glossy Photo"
            }
          ]
        }
      }
    ],
    "color": {
      "option": [
        {
          "vendor_id": "psk:Color",
          "type": "STANDARD_COLOR",
          "is_default": true
        },
        {
          "vendor_id": "psk:Monochrome",
          "type": "STANDARD_MONOCHROME",
        }
      ]
    },
    "duplex": {
      "option": [
        {
          "type": "NO_DUPLEX",
          "is_default": true
        },
        { "type": "LONG_EDGE" },
        { "type": "SHORT_EDGE" }
      ]
    },
    "page_orientation": {
      "option": [
```

```
    <psf:Value xsi:type="xsd:string">Color</psf:Value>
  </psf:Property>
  <psf:Option name="psk:Color" constrained="psk:None">
    <psf:Property name="psk:DisplayName">
      <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>
```

```
        {
          "type": "PORTRAIT",
          "is_default": true
        },
        { "type": "LANDSCAPE" }
      ]
    },
    "copies": {
      "default": 1,
      "max": 999
    },
    "dpi": {
      "option": [
        {
          "horizontal_dpi": 300,
          "vertical_dpi": 300,
          "is_default": true
        },
        {
          "horizontal_dpi": 600,
          "vertical_dpi": 600
        }
      ]
    },
    "fit_to_page": {
      "option": [
        {
          "type": "NO_FITTING",
          "is_default": true
        },
        { "type": "FIT_TO_PAGE" }
      ]
    },
    "media_size": {
      "option": [
        {
          "name": "NA_LETTER",
          "width_microns": 215900,
          "height_microns": 279400,
          "is_default": true
        },
        {
          "name": "NA_LEGAL",
          "width_microns": 215900,
          "height_microns": 355600
        },
        {
          "name": "ISO_A5",
          "width_microns": 148000,
          "height_microns": 210000
        },
        {
          "name": "ISO_A4",
          "width_microns": 210000,
          "height_microns": 297000
        },
        {
          "name": "CUSTOM",
          "width_microns": 55000,
          "height_microns": 91000,
          "custom_display_name":
            "Card 2.16”x3.58” 55x91mm"
        },
        {
          "name": "ISO_A3",
          "width_microns": 297000,
          "height_microns": 420000
        },
        {
          "name": "ISO_A2",
          "width_microns": 420000,
          "height_microns": 594000
```

```
 <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>
 <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>
```

```
        }
      ]
    },
    "collate": { "default": false },
    "reverse_order": { "default": false }
  }
}
```

```
    </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">
  <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

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

  // Content types (sometimes referred to as MIME types) that are supported by
  // the printer.
  //
  // The order of these types determines which content type the document should
  // be converted to. For example, if the types are ordered as:
  //
  //   [
  //     {"content_type": "application/pdf"},
  //     {"content_type": "image/pwg-raster"}
  //   ]
  //
  // Then the document's content type will first be matched to any content type
  // in the list. If there is a match, then the document will be sent to the
  // printer as is. If there is no match, then the document will be converted to
  // a content type which the server supports starting from the first option. In
  // this example, if the document is sent as "text/html" and the printer
  // supports "application/pdf" and "image/pwg-raster", then the document will
  // be converted to "application/pdf" and not "image/pwg-raster", because
  // "application/pdf" is declared earlier in this list.
  repeated 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;

  // Physical model of the printer's input trays.
  repeated InputTrayUnit input_tray_unit = 4;

  // Physical model of the printer's output bins.
  repeated OutputBinUnit output_bin_unit = 5;

  // Physical model of the printer's markers.
  repeated Marker marker = 6;

  // Physical model of the printer's covers.
  repeated Cover cover = 7;

  // Physical model of the printer's media paths.
  repeated MediaPath media_path = 8;

  // Vendor-provided printer capabilities.
  repeated VendorCapability vendor_capability = 101;

  // Color printing capabilities of the printer.
```

```
  optional Color color = 102;

  // Duplexing capabilities of the printer.
  optional Duplex duplex = 103;

  // Page/paper orientation capabilities of the printer.
  optional PageOrientation page_orientation = 104;

  // Multiple copy capability of the printer.
  optional Copies copies = 105;

  // Page margins capability of the printer.
  optional Margins margins = 106;

  // Printing quality or dots-per-inch (DPI) capabilities of the printer.
  optional Dpi dpi = 107;

  // Page fitting capabilities of the printer.
  optional FitToPage fit_to_page = 108;

  // Page range selection capability of the printer.
  optional PageRange page_range = 109;

  // Page or media size capabilities of the printer.
  optional MediaSize media_size = 110;

  // Paper collation capability of the printer.
  optional Collate collate = 111;

  // Reverse order printing capability of the printer.
  optional ReverseOrder reverse_order = 112;
}

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

  // Minimum supported version of the content type if applicable (e.g. "1.5").
  optional string min_version = 2;

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

// 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
  //
  //   {f
  //     "speed_ppm": 5.5,
  //     "color_type": ["STANDARD_MONOCHROME"],
  //     "media_size_name": ["NA_LETTER", "ISO_A4"]
```

```
  //   }
  //
  // indicates that the printer prints at 5.5 pages per minute when printing in
  // STANDARD_MONOCHROME in either NA_LETTER or ISO_A4 paper sizes.
  message Option {
    // Speed measured in pages per minute (required).
    optional float speed_ppm = 1;

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

    // Names of media sizes that operate at this speed.
    repeated MediaSize.Name media_size_name = 3;

  }

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

// Configuration of how printer should receive PWG raster images.
message PwgRasterConfig {
  // Transformation to apply to pages during PWG rasterization.
  message Transformation {
    // Types of transformation operations to apply.
    enum Operation {
      // Rotate pages 180 degrees.
      ROTATE_180 = 0;

      // Flip pages along the long edge of the paper.
      FLIP_ON_LONG_EDGE = 1;

      // Flip pages along the short edge of the paper.
      FLIP_ON_SHORT_EDGE = 2;
    }

    // Selectors of which pages to apply the transformation to.
    enum Operand {
      // Apply transformation to all pages.
      ALL_PAGES = 0;

      // Apply transformation to even pages only when duplexing.
      ONLY_DUPLEXED_EVEN_PAGES = 1;

      // Apply transformation to odd pages only when duplexing.
      ONLY_DUPLEXED_ODD_PAGES = 2;
    }

    // Required.
    optional Operation operation = 1;

    // Required.
    optional Operand operand = 2;
  }

  // What transformations to apply to pages in the print job.
  repeated Transformation transformation = 1;
}
```

```
// Physical model of a printer input tray.
message InputTrayUnit {
  // Enumeration of input tray types.
  enum Type {
    CUSTOM = 0;
    INPUT_TRAY = 1;
    BYPASS_TRAY = 2;
    MANUAL_FEED_TRAY = 3;
    LCT = 4; // Large capacity tray.
    ENVELOPE_TRAY = 5;
    ROLL = 6;
  }

  // Vendor-provided ID of the input tray (required).
  optional string vendor_id = 1;

  // Type of input tray (required).
  optional Type type = 2;

  // Index of the input tray.
  optional int64 index = 3;

  // Custom display name of the input tray. Only needed for CUSTOM tray type.
  optional string custom_display_name = 4;
}

// Physical model of a printer output bin.
message OutputBinUnit {
  // Enumeration of output bin types.
  enum Type {
    CUSTOM = 0;
    OUTPUT_BIN = 1;
    MAILBOX = 2;
    STACKER = 3;
  }

  // Vendor-provided ID of the output bin (required).
  optional string vendor_id = 1;

  // Type of output bin (required).
  optional Type type = 2;

  // Index of output bin.
  optional int64 index = 3;

  // Custom display name of the output bin. Only needed for CUSTOM bin type.
  optional string custom_display_name = 4;
}

// Physical model of a printer marker.
message Marker {
  // Enumeration of types of printer markers.
  enum Type {
    CUSTOM = 0;
    TONER = 1;
    INK = 2;
    STAPLES = 3;
  }
```

```
  // Message that describes the color of a marker.
  message Color {
    // Enumeration of color types of the printer marker.
    enum Type {
      CUSTOM = 0;
      BLACK = 1;
      COLOR = 2;
      CYAN = 3;
      MAGENTA = 4;
      YELLOW = 5;
      LIGHT_CYAN = 6;
      LIGHT_MAGENTA = 7;
      GRAY = 8;
    }

    // Required.
    optional Type type = 1;

    // Custom display name of the color. Only needed with CUSTOM color type.
    optional string custom_display_name = 2;
  }

  // Vendor-provided ID of the marker (required).
  optional string vendor_id = 1;

  // Type of marker (required).
  optional Type type = 2;

  // Color of the marker. Only needed if marker type is INK or TONER.
  optional Color color = 3;

  // Custom display name of the marker. Only needed with CUSTOM marker type.
  optional string custom_display_name = 4;
}

// Physical model of a printer cover.
message Cover {
  // Enumeration of cover types.
  enum Type {
    CUSTOM = 0;
    DOOR = 1;
    COVER = 2;
  }

  // Vendor-provided ID of the cover (required).
  optional string vendor_id = 1;

  // Type of the cover (required)
  optional Type type = 2;

  // Index of the cover.
  optional int64 index = 3;

  // Custom display name of the cover. Only needed with CUSTOM cover type.
  optional string custom_display_name = 4;
}

// Physical model of a media path of a printer. Media paths are the paths
// through which print media flows.
```

```
message MediaPath {
  // Vendor-provided ID of a media path (required).
  optional string vendor_id = 1;
}

// 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;
  }

  message Option {
    optional Type type = 1 [default = NO_DUPLEX];
    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;
    AUTO = 2;
  }

  message Option {
    // Type of page orientation (required).
    optional 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 {
  // Enumerates the set of predefined types of margins. Devices should use
  // these types to semantically describe the margins option. This type will
  // be used for UI purposes only.
  enum Type {
    BORDERLESS = 0;
    STANDARD = 1;
    CUSTOM = 2;
  }

  message Option {
```

```
    // Type of margin option (required).
    optional Type type = 1;

    // Top margin of the page (required).
    optional int32 top_microns = 2;

    // Right margin of the page (required).
    optional int32 right_microns = 3;

    // Bottom margin of the page (required).
    optional int32 bottom_microns = 4;

    // Left margin of the page (required).
    optional int32 left_microns = 5;

    optional bool is_default = 6 [default = false];
  }

  repeated Option option = 1;
}

// Capability that defines a set of page fitting options available on a device.
message FitToPage {
  // Enumeration of page fitting algorithms. The "page" is defined as the media
  // size minus any given margins.
  enum Type {
    NO_FITTING = 0;
    FIT_TO_PAGE = 1;
    GROW_TO_PAGE = 2;
    SHRINK_TO_PAGE = 3;
    FILL_PAGE = 4;
  }

  message Option {
    // Type of fitting algorithm (required).
    optional Type type = 1;
    optional bool is_default = 2 [default = false];
  }

  repeated Option option = 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).
    optional 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 {
  optional bool default = 1 [default = true];
}

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

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

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

// Ticket item indicating which page orientation option to use.
message PageOrientationTicketItem {
  // Page orientation type (required).
  optional PageOrientation.Type type = 1;
}

// Ticket item indicating how many copies to produce.
message CopiesTicketItem {
  // Number of copies to print (required).
  optional int32 copies = 1;
}

// Ticket item indicating what margins to use (in microns).
message MarginsTicketItem {
  // Top margin of the page (required).
  optional int32 top_microns = 1;

  // Top margin of the page (required).
  optional int32 right_microns = 2;

  // Top margin of the page (required).
  optional int32 bottom_microns = 3;

  // Top margin of the page (required).
  optional int32 left_microns = 4;
}
```

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

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

// Ticket item indicating whether to collate pages.
message CollateTicketItem {
  // Whether to print collated (required).
  optional bool collate = 1;
}

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

5.2 Scanner Section Objects

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

  // Vendor-provided printer capabilities.
  repeated VendorCapability vendor_capability = 101;

  // Color scanning capabilities of the scanner.
  optional Color color = 102;

  // Image quality or dots-per-inch capabilities of the scanner.
  optional Dpi dpi = 103;

  // Image size capabilities of the scanner.
  optional MediaSize media_size = 104;

  // File format type capabilities of the scanner.
  optional FileFormat file_format = 105;
}

// 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 {
```

```
    // Type of the file format (required).
    optional 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 {
  repeated VendorTicketItem vendor = 1;
  optional ColorTicketItem color = 2;
  optional DpiTicketItem dpi = 3;
  optional MediaSizeTicketItem media_size = 4;
  optional FileTypeTicketItem file_type = 5;
}

// Ticket item indicating what pages to use.
message FileTypeTicketItem {
  // Type of the file format (required).
  optional FileFormat.Type type = 1;

  // Custom content type if applicable (required iff type == CUSTOM).
  optional string custom_content_type = 2;
}
```

5.3 Common Objects

```
// 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).
  optional string id = 1;

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

  // Type of this capability (required).
  optional 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;
}

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

  // Data type of the value of the range capability (required).
  optional ValueType value_type = 1;
  optional string default = 2;
  optional string min = 3;
  optional string max = 4;
}

// 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).
    optional string value = 1;

    // User-friendly string to represent this option (required).
    optional 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;
}

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

  // Type of data of the typed value capability (required).
  optional ValueType value_type = 1;
  optional string default = 2;
}
```

```
// 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;
    AUTO = 4;
  }

  message Option {
    // ID to help vendor identify the color option.
    optional 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.
    // This field is required.
    optional 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 {
  message Option {
    // Horizontal DPI (required).
    optional int32 horizontal_dpi = 1;

    // Vertical DPI (required).
    optional int32 vertical_dpi = 2;

    optional bool is_default = 3 [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 {
  // Enumeration of media size names. This is used for UI purposes.
  enum Name {
    CUSTOM = 0;
```

```
    // North American standard sheet media names.
    NA_INDEX_3X5 = 100;
    NA_PERSONAL = 101;
    NA_MONARCH = 102;
    NA_NUMBER_9 = 103;
    NA_INDEX_4X6 = 104;
    NA_NUMBER_10 = 105;
    NA_A2 = 106;
    NA_NUMBER_11 = 107;
    NA_NUMBER_12 = 108;
    NA_5X7 = 109;
    NA_INDEX_5X8 = 110;
    NA_NUMBER_14 = 111;
    NA_INVOICE = 112;
    NA_INDEX_4X6_EXT = 113;
    NA_6X9 = 114;
    NA_C5 = 115;
    NA_7X9 = 116;
    NA_EXECUTIVE = 117;
    NA_GOVT_LETTER = 118;
    NA_GOVT_LEGAL = 119;
    NA_QUARTO = 120;
    NA_LETTER = 121;
    NA_FANFOLD_EUR = 122;
    NA_LETTER_PLUS = 123;
    NA_FOOLSCAP = 124;
    NA_LEGAL = 125;
    NA_SUPER_A = 126;
    NA_9X11 = 127;
    NA_ARCH_A = 128;
    NA_LETTER_EXTRA = 129;
    NA_LEGAL_EXTRA = 130;
    NA_10X11 = 131;
    NA_10X13 = 132;
    NA_10X14 = 133;
    NA_10X15 = 134;
    NA_11X12 = 135;
    NA_EDP = 136;
    NA_FANFOLD_US = 137;
    NA_11X15 = 138;
    NA_LEDGER = 139;
    NA_EUR_EDP = 140;
    NA_ARCH_B = 141;
    NA_12X19 = 142;
    NA_B_PLUS = 143;
    NA_SUPER_B = 144;
    NA_C = 145;
    NA_ARCH_C = 146;
    NA_D = 147;
    NA_ARCH_D = 148;
    NA_ASME_F = 149;
    NA_WIDE_FORMAT = 150;
    NA_E = 151;
    NA_ARCH_E = 152;
    NA_F = 153;

    // Chinese standard sheet media size names.
    ROC_16K = 200;
```

```
    ROC_8K = 201;
    PRC_32K = 202;
    PRC_1 = 203;
    PRC_2 = 204;
    PRC_4 = 205;
    PRC_5 = 206;
    PRC_8 = 207;
    PRC_6 = 208;
    PRC_3 = 209;
    PRC_16K = 210;
    PRC_7 = 211;
    OM_JUURO_KU_KAI = 212;
    OM_PA_KAI = 213;
    OM_DAI_PA_KAI = 214;
    PRC_10 = 215;

    // ISO standard sheet media size names.
    ISO_A10 = 301;
    ISO_A9 = 302;
    ISO_A8 = 303;
    ISO_A7 = 304;
    ISO_A6 = 305;
    ISO_A5 = 306;
    ISO_A5_EXTRA = 307;
    ISO_A4 = 308;
    ISO_A4_TAB = 309;
    ISO_A4_EXTRA = 310;
    ISO_A3 = 311;
    ISO_A4X3 = 312;
    ISO_A4X4 = 313;
    ISO_A4X5 = 314;
    ISO_A4X6 = 315;
    ISO_A4X7 = 316;
    ISO_A4X8 = 317;
    ISO_A4X9 = 318;
    ISO_A3_EXTRA = 319;
    ISO_A2 = 320;
    ISO_A3X3 = 321;
    ISO_A3X4 = 322;
    ISO_A3X5 = 323;
    ISO_A3X6 = 324;
    ISO_A3X7 = 325;
    ISO_A1 = 326;
    ISO_A2X3 = 327;
    ISO_A2X4 = 328;
    ISO_A2X5 = 329;
    ISO_A0 = 330;
    ISO_A1X3 = 331;
    ISO_A1X4 = 332;
    ISO_2A0 = 333;
    ISO_A0X3 = 334;
    ISO_B10 = 335;
    ISO_B9 = 336;
    ISO_B8 = 337;
    ISO_B7 = 338;
    ISO_B6 = 339;
    ISO_B6C4 = 340;
    ISO_B5 = 341;
    ISO_B5_EXTRA = 342;
```

```
    ISO_B4 = 343;
    ISO_B3 = 344;
    ISO_B2 = 345;
    ISO_B1 = 346;
    ISO_B0 = 347;
    ISO_C10 = 348;
    ISO_C9 = 349;
    ISO_C8 = 350;
    ISO_C7 = 351;
    ISO_C7C6 = 352;
    ISO_C6 = 353;
    ISO_C6C5 = 354;
    ISO_C5 = 355;
    ISO_C4 = 356;
    ISO_C3 = 357;
    ISO_C2 = 358;
    ISO_C1 = 359;
    ISO_C0 = 360;
    ISO_DL = 361;
    ISO_RA2 = 362;
    ISO_SRA2 = 363;
    ISO_RA1 = 364;
    ISO_SRA1 = 365;
    ISO_RA0 = 366;
    ISO_SRA0 = 367;

    // Japanese standard sheet media size names.
    JIS_B10 = 400;
    JIS_B9 = 401;
    JIS_B8 = 402;
    JIS_B7 = 403;
    JIS_B6 = 404;
    JIS_B5 = 405;
    JIS_B4 = 406;
    JIS_B3 = 407;
    JIS_B2 = 408;
    JIS_B1 = 409;
    JIS_B0 = 410;
    JIS_EXEC = 411;
    JPN_CHOU4 = 412;
    JPN_HAGAKI = 413;
    JPN_YOU4 = 414;
    JPN_CHOU2 = 415;
    JPN_CHOU3 = 416;
    JPN_OUFUKU = 417;
    JPN_KAHU = 418;
    JPN_KAKU2 = 419;

    // Other metric standard sheet media size names.
    OM_SMALL_PHOTO = 500;
    OM_ITALIAN = 501;
    OM_POSTFIX = 502;
    OM_LARGE_PHOTO = 503;
    OM_FOLIO = 504;
    OM_FOLIO_SP = 505;
    OM_INVITE = 506;
  }

  message Option {
```

```
    optional Name name = 1 [default = CUSTOM];
    optional int32 width_microns = 2;
    optional int32 height_microns = 3;
    optional bool is_continuous_feed = 4;
    optional bool is_default = 5 [default = false];

    // Should only be set if the "name" field is CUSTOM.
    optional string custom_display_name = 6;
  }

  repeated Option option = 1;
  optional int32 max_width_microns = 2;
  optional int32 max_height_microns = 3;
  optional int32 min_width_microns = 4;
  optional int32 min_height_microns = 5;
}

// Ticket item indicating which color option to use.
message ColorTicketItem {
  optional string vendor_id = 1;

  // Type of the color (required).
  optional Color.Type type = 2;
}

// Ticket item indicating what image resolution to use.
message DpiTicketItem {
  // Horizontal DPI to print at (required).
  optional int32 horizontal_dpi = 1;

  // Vertical DPI to print at (required).
  optional int32 vertical_dpi = 2;
}

// Ticket item indicating what media size to use.
message MediaSizeTicketItem {
  // Width (in microns) of the media to print to.
  optional int32 width_microns = 1;

  // Height (in microns) of the media to print to.
  optional int32 height_microns = 2;

  // Whether the media size selection is continuously fed. If false, both width
  // and height must be set. If true, only one need be set.
  optional bool is_continuous_feed = 3 [default = false];
}
```

5.4 Integration with GCP

The data structures used to communicate CDDs and CJTs are defined in Protobuf format; however, 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 alongside 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. Make sure to include the “use_cdd=true” parameter when registering a printer capabilities in CDD format.

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 CDD format, add the “use_cdd=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 depending on the “format” parameter. However, no longer do devices need to understand either of these formats since now they only need to understand the CJT format. To get a CJT for a given print job, use the /ticket?jobid=<jobid>&use_cjt=true API. Here’s an example of the response of a /ticket?jobid=1234&use_cjt=true request (formatted and commented for convenience):

```
{
  "version": "1.0",
  "print": {
    "vendor_ticket_item": [],
    "color": {
      "vendor_id": "grayscale",
      "type": "STANDARD_MONOCHROME"
    },
    "copies": { "copies": 3 }
  }
}
```

6. Changelog

This section lists the **incompatible** changes in this document since June 3, 2013.

1. Changed the name of the field “door” of PrinterDescriptionSection to “cover”.
2. Added the field “custom_display_name” to Marker.Color which is now required when the color type is “CUSTOM”.
3. Changed data structure of “supported_content_type” field of PrinterDescriptionSection. Removed “rank” field. Now order is determined by order of the elements in the array.