CIM Printer Model

Whitepaper

Draft 0.4

26 May 1999

Please send all comments to davidsi@sco.com

1. Objectives

The high level objective of providing a printer model within CIM is to show how a printer, its jobs and any controlling systems fit into the managed environment. The logical model shows how the printer is, or could be, configured and how the print capabilities it provides may be accessed. The model does not take over the printing functionality or the protocols that are necessary to access a printer, but rather describes how such mechanisms can be managed.

CIM V2.1 already includes a Printer object (CIM_Printer) so the new CIM 2.2 printer object (and other related objects) discussed in this paper build on this existing definition. In particular it was a requirement for this extension, that there was complete backwards compatibility with the existing printer based definitions. In some cases, this existing representation was different to the approach taken by alternative models, such as that used by IPP, and in such cases the need to maintain backwards compatibility has taken priority.

The intent of the model is that it is applicable to any possible real-world realisation of a printer and its supporting software such as drivers, spoolers etc. Where necessary, the model assumes a generic approach so that if necessary a vendor can derive new classes that provide behaviours that are specific to their products.

1.1 Relationship to Existing MIBs

Several MIBs have been defined to provide a standard means of instrumenting printers, their jobs and the various printing options. These include:

Printmib-mib-info (draft-ietf-printmib-finishing-08.txt)
Printmib-job-monitor (draft-ietf-printmib-job-monitor-07.txt)
Printmib-finishing (draft-ietf-print-mib-info-04.txt)

Data from these MIBs can be mapped into the CIM hierarchy. Their mapping is indicated using the MappingStrings qualifier on a property.

1.2 Relationship to IPP

Internet Printing Protocol (IPP) provides a large range of attributes spread across several objects:

- Printer
- Jobs
- Job Template

When compared to the CIM representation it can be seen that the IPP printer representation encompasses not just the printer device (e.g. a mechanism which renders onto media) but also the logic that manages jobs waiting to use that device and the interpretation of the requests embedded in a job. Although this contrasts with the relatively simple definition of a Printer that CIM uses, it is entirely consistent with the overall CIM printer model. The CIM model chooses to break out the functionality that IPP represents in a printer into Printer, PrintQueue, PrintService and PrintSAP objects. Print jobs are addressed via subclassing of the CIM_Job object.

As a result of the additional objects that are present in the CIM model, mappings between the IPP representation and the CIM representation may require traversal of one or more associations in the CIM version to locate the right object with the appropriate attribute. In general the mapping is 1-to-1 between IPP and CIM. There are some situations where the mapping may be more complex and is potentially 1-to-n. These scenarios occur with the IPP attributes such as printer-name where several CIM objects have a name property that could be used in its place. For attributes such as this, some additional semantic interpretation of the IPP property is necessary, for example printer-name could be interpreted as the name of a queue exported by a controlling system as that is the point of contact for a user wishing to print a job. Although this requires additional effort when instantiating a CIM representation of IPP information, the models are compatible and relevant mappings can be established.

All references to IPP in this document are based upon RFC 2566 (“Internet Printing Protocol/1.0: Model and Semantics” April 1999).

2. Printer Management Objects

Any model for the management of a printer and its related infrastructure must take into account the different types of printing topologies that might be deployed within an enterprise. The simplest scenario consists of a printer that is directly attached to a computer system that is not networked, only supports a single user at a time and is not running any spooling software. Adding a print spooler may provide additional management capabilities for a print job such as job prioritization. More complex scenarios arise when systems are networked together. In such cases each client may be able to manage print jobs as well as the networked entity that is controlling the print mechanism. In other deployments the printer may be a “black box” where it simply provides an IP address to which clients can send jobs that are to be output, or the printer may be another instance of a generic computer system with a directly attached printer.

The model is a logical representation for printing. It is not intended to cover the physical aspects of a printer such as the form factor for any of the printer components, asset information and power information. The physical model within CIM already provides the necessary objects and associations that allow this type of information to specified within the CIM namespace and the relevant physical entities to be associated with the logical functionality that they provide. An example physical use case is discussed in this document in section 6. Figure 1 illustrates the main objects in the printer model.

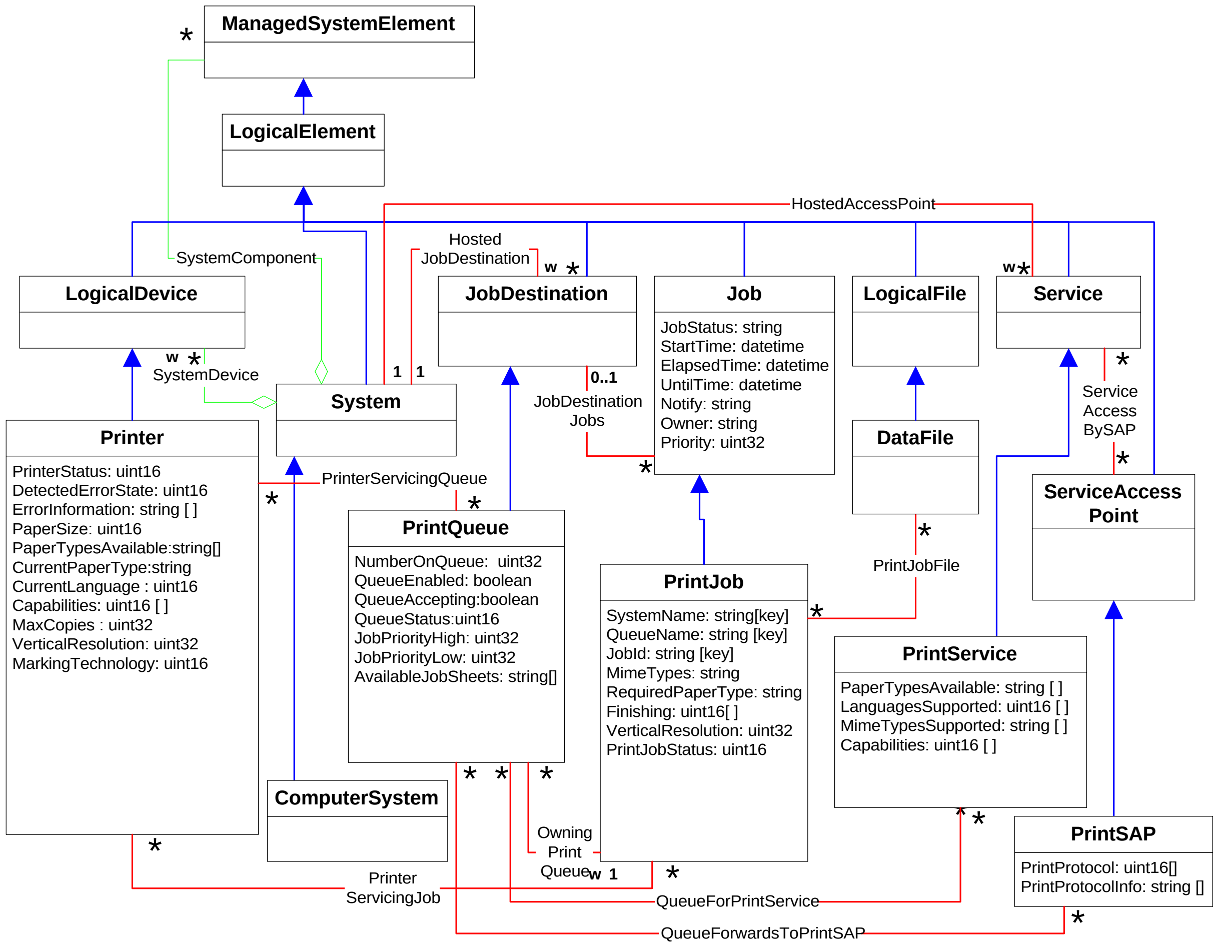

Figure 1. High level overview of the printer related objects and associations.

The key objects that the printer model provides are:

Printer

The printer takes print jobs and renders them onto physical media such as paper. The areas of management interest consist of the current state of the printer (e.g. accepting jobs and possible error state), the paper sizes and types available, the page description languages (PDLs) it uses and other capabilities such as duplex printing and stapling.

PrintQueue

The print queue is used to contain and marshal print jobs. A print queue may be a very sophisticated entity that allows/enables job prioritization and job transformation or it might be very simple with no real manageable properties. The simplest type of print queue might be the FIFO functionality that is provided by a device driver that simply outputs a byte stream to a parallel or serial device.

PrintJob

In general terms a print job is a byte stream that represents the image to be rendered and the meta information that describes how it should be managed by the print subsystem and the characteristics/requirements of the job. In the CIM model the print job object only describes the meta information that relates to the job, with the byte stream being optionally represented as a file that is associated with the PrintJob object.

PrintService

The print service provides the functionality that manages print requests and the printer. It could be viewed as the system process or daemon that provides printing capabilities and any related binaries that are executed to obtain print service.

PrintSAP

The print service access point describes how the print service is accessed for example it describes the printing protocols that may be supported by the underlying print service.

In addition to these objects, other objects may play a key role in defining the management of a printer and its controlling software:

System and ComputerSystem

These objects can be used to show how the Printer is related to a system and identify the host for the print services. In some cases the ComputerSystem may be a “general purpose” computer that is also acting as a host for a printer that is either directly attached or reached via the network. In other cases the ComputerSystem may represent the additional manageable characteristics of a stand-alone printer, e.g in the case of a stand-alone printer that is network enabled and can field requests from any other entity on the network.

DataFile

Used to represent the information that will be sent to the printer.

The relationship between the objects is summarized in figure 1.

The remaining portion of this section discusses the printer specific objects in more detail.

2.1 Printer

The Printer object defines the following attributes.

`PrinterStatus: uint16`

Describes the current state of a printer. The suggested interpretation of the status is:

`Other`	Compatibility use only
`Unknown`	The state of the printer cannot be determined
`Idle`	Waiting for a print job
`Printing`	Printing a print job
`Warmup`	Warming up. May transition to Idle, Printing or Stopped Printing.
`Stopped Printing`	The printer has stopped either in the middle of a Job or there are other jobs outstanding that cannot be processed because of an error etc.
`Offline`	The printer has been taken off-line and when put back on-line may be either in the “Stopped Printing”, “Idle” or “Printing States”.

`DetectedErrorState : uint16`

Indicates the current error condition for the printer. Values include “No Error", "Low Paper" etc.

The use of “Offline” as defined in CIM2.1 is DEPRECATED by this revision. Instances that wish to represent an off-line printer are recommended to set PrinterStatus to “Offline” and provide any additional information in DetectedErrorState.

`ErrorInformation : string []`

This may be used to provide additional error information.

`PaperSizesSupported : uint16[]`

CIM 2.1 defines this as an integer array indicating the types of paper that can be physically supported by the printer. Values include "Letter", "Legal", "NA-10x13-Envelope", "NA-9x12-Envelope", etc. For CIM 2.2 an additional property PaperTypesAvailable has been added to express the paper types that are available to a user of the printer.

`PaperTypesAvailable: string []`

This provides descriptions of the paper sizes and colors that are available. The strings should be formed as specified by DPA (ISO19175) and the Printer MIB (RFC 1539). Typically the strings listed for this property indicate the attributes of the paper that have been currently loaded into the printer. However in the case of some implementations there may be paper sizes/colors listed that require manual intervention (e.g. the print service may issue a request to an administrator to change paper as part of processing a request).

`DefaultPaperType: string`

This specifies the paper size and color that will be used by the printer if a PrintJob does not otherwise specify a requirement. The string should be formed as specified by DPA (ISO19175) and the Printer MIB (RFC 1539).

`CurrentPaperType: string`

This specifies the paper size and color that is currently being used by the printer. The string should be formed as specified by DPA (ISO19175) and the Printer MIB (RFC 1539).

`LanguagesSupported : uint16[]`

An array indicating the print languages natively supported. For example "PCL", "HPGL", "PJL", "PS" etc.

An addition to the model from CIM 2.1 is to include a new value "Mime". If this is set then the MimeTypesSupported property should describe which mime types the printer will support. Both mime and regular PDLs may be specified.

`MimeTypesSupported: string []`

This is a new addition since CIM 2.1. It allows mime types to be specified that the printer can interpret. If values are set in this property "Mime" should be set in LanguagesSupported.

`CurrentLanguage : uint16`

Specifies the PDL that the printer is currently using to interpret the active print job or if no print job is being processed the current state of the printer for the next job. If the printer will use mime information then this should be set to "Mime" and the appropriate mime types set in CurrentMimeType.

`CurrentMimeType: string`

The mime type of the job currently being processed if LanguageSetting is set to "Mime"

`DefaultLanguage : uint16`

Specifies the PDL that will be used to interpret a new print job if no other PDL is specified. If the printer will use mime information then this should be set to "Mime" and the appropriate mime types set in `DefaultMimeType`.

`DefaultMimeType: string`

The default mime type that the printer will use for interpreting a printjob if DefaultLanguage is set to "Mime".

`JobCountSinceLastReset: uint32`

The number of printer jobs processed since last reset. These jobs may be "processed from one or more PrintQueue queues.

`TimeOfLastReset: datetime`

The time of last reset of the Printer Device. For a printer that is directly attached to a ComputerSystem, e.g. via a serial connection, this could be the time at which that system was last booted.

`Capabilities: uint16[]`

This specifies how the document should be finished and the requirements it has on the printer. The values it provides map onto the Capabilities provided by a printer. For CIM 2.1 these were defined as:

```
{"Unknown", "Other", "Color Printing",
             "Duplex Printing", "Copies", "Collation", "Stapling",
             "Transparency Printing"},
```

"Copies" may be specified in Capabilities but an instance of a Printer that supports multiple copies being created on the output device should set the MaxCopies and DefaultCopies properties appropriately.

The following additional Capabilties have been added beyond those present in CIM 2.1:

`Punch`	Finish by punching holes
`Cover`	Finish with covers
`Bind`	Bind
`BlackWhite Printing`	Used to indicate that a color printer has an alternative mechanism for printing black and white.
`OneSided`	Duplex settings (The CIM 2,1 “Duplex” is equivalent to ipp “two sided”)
`TwoSidedLongEdge`	
`TwoSidedShortEdge`	
`Portrait`	Page orientations
`Landscape`	
`Reverse Portrait`	
`Reverse Landscape`	
`Quality High`	High quality printing
`Quality Normal`	Normal quality printing
`Quality Low`	Low quality output

`DefaultCapabilities: uint16[]`

Describes the capabilities of the printer that will be applied to a job if that job does not specify otherwise.

`CurrentCapabilities: uint16[]`

Describes the capabilities of the printer which are currently being used.

`CapabilityDescriptions: string []`

An array of strings providing more detailed explanations for any of the Printer features indicated in the Capabilities array. Each entry of this array is related to the entry in the Capabilities array that is located at the same index.

`MaxCopies : uint32`

The maximum number of copies that the printer can provide for a single PrintJob.

`DefaultCopies : uint32`

The number of copies that the printer will produce for a single PrintJob if no explicit request is made for a number of copies.

`MaxNumberUp : uint32`

The maximum number of print-stream pages that the printer can render on a single side of medium.

`DefaultNumberUp : uint32`

The number of print-stream pages that the printer will render on a single side of medium unless the print job specifies a different value.

`HorizontalResolution: uint32`

The Printer's horizontal resolution in Pixels per Inch

`VerticalResolution: uint32`

The Printer's vertical resolution in Pixels per Inch

`CharSetsSupported : string []`

Identifies the available character sets for management text used by the printer. Strings specified in this property should conform to the semantics and syntax specified by RFC 2046 and contained in the IANA character-set Registry. Examples include “utf-8”, “iso-8859-1”.

`CurrentCharSet : uint16`

The current character set being used by the printer. This value provides an index into the array specified for the CharSetsSupported property.

`NaturalLanguageSupported : string []`

Identifies the available languages for textual strings used in the management of the printer. The values for strings contained in these strings are specified by RFC 1766. Examples include “en” for English and “de” for German.

`CurrentNaturalLanguage : uint16`

Identifies the language for any textual strings used in the management of the printer.

`MaxSizeSupported : uint32`

The largest byte stream (expressed as Kbytes) that this printer will accept as a single job.

`AvailableJobSheets : string[]`

Describes the job sheets that are supported by the printer.

`MarkingTechnology : uint16`

Specifies the type of mechanism use to render on to the media (e.g. laser, dot matrix, inkjet, e.t.c).

2.2 PrintQueue

PrintQueue is derived from JobDestination and as such it is used to refer to a queue that may contain zero or more jobs. A print job that is on a queue may be in one of a number of states:

- "Pending" - Waiting for a destination printer to become available
- "Printing" - Having reached the head of the queue, the print job is being printed or is being transferred to another PrintService which will manage it from this point on.
- "Complete" - The print job has been processed by this queue (e.g. it has either been printed out, passed to another print service or passed to another queue within the current print service). No further processing of the job will occur.

In many cases PrintJobs that have been completed may not remain as instantiated objects that are scoped by the print queue. However there are situations when implementers may choose to model this information:

1. Auditing purposes. The completed jobs state allows a management application to provide a complete log of who submitted jobs to a queue, the characteristics of those jobs and when they were processed.
2. Job Tracking. In a multi-tiered environment (a queue passes jobs to another print service) the job may not appear on physical media until another service has completed its processing and it is appropriate to track the queues that the job has passed through until the job has been handled by a printer.

PrintJobs waiting on the queue can be located using the OwningPrintQueue association. This association is used to scope the PrintJob (PrintJob is weak to PrintQueue). The Printer(s) that will service the PrintJobs are determined by following the PrinterServicingQueue association. When a queue forwards requests to a PrintService, the association QueueForwardsToPrintSAP indicates the destination print service.

The PrintQueue object provides relatively few properties as most of the configuration either occurs on individual PrintJobs or at the Printer itself. Most of the properties are limited to describing the queue's behaviour in processing or managing PrintJobs, for example whether it will accept a job or pass a job to a printer. An exception is MaxJobSize that can specify the largest possible job that a queue might accept. This is provided to cover the scenario where a queue provides spooling functionality by taking a local copy of the data stream that is to be passed to the Printer. In such cases, to prevent disks filling up or a denial of service attack, an administrator may wish to indicate that only jobs less than a particular size are accepted by the queue. It should be noted that some queues may provide a spooling ability by simply tracking the original location of the data stream rather than taking a copy and in such cases limiting the job size at the queue may not be necessary.

The additional attributes that are defined by PrintQueue are:

`QueueEnabled:boolean`
: When TRUE the PrintService is taking jobs from the PrintQueue and passing them to the appropriate Printer or PrintService. If FALSE, the PrintQueue will not pass jobs to any of its associated Printers or PrintServices and the queue may grow if jobs are passed to it.

`QueueAccepting:boolean`
: When TRUE the PrintService can accept jobs through the PrintSAP and place them on the PrintQueue. If the PrintService has not been started then jobs cannot be queued.

`NumberOnQueue:uint32`
: Number of jobs on the print queue.

`QueueStatus:uint32`
: General status can be indicated in ManagedSystemElement.status with this attribute being used to provide any additional warning or error conditions that are related to the queue:

  `Other`
  : A specific status applies to the queue but cannot be expressed by the QueueStatus enumeration. It is recommended that QueueStatusInfo is always assigned a useful string when assigning other to QueueStatus.

  `Unknown`
  : The status cannot be determined

  `QueueSpoolAreaFull`
  : The storage used by the queue to hold pending jobs is exhausted.

  `QueueStopping`

The value ”QueueStopping” may be assigned if the queue is soon to be stopped but for the moment is continuing to accept and process requests.

`QueueStatusInfo: string`

Additional information to describe the current status of the print queue.

`MaxJobSize: uint32`

Specifies the largest size of job that may be submitted to the queue. A value of zero indicates unlimited. In the future additional policies may be used to control the size of job that can be submitted by an entity to the queue.

`JobPriorityDefault: uint32`

Specifies the default priority that will be assigned to a job that is added to the queue when no other priority is specified.

`JobPriorityHigh: uint32`

Specifies the numeric value that is assigned to jobs of the highest priority (most important). It should be noted that it is reasonable to instantiate a queue where the JobPriorityHigh is numerically less than JobPriorityLow as this indicates that higher priority jobs are represented by lower numeric values. If JobPriorityHigh and JobPriorityLow are assigned to zero then it can be assumed the print queue does not support job prioritization.

`JobPriorityLow: uint32`

Specifies the numeric value that is assigned to jobs of the lowest priority (least important). When JobPriorityLow is assigned a value less than JobPriorityHigh then jobs with higher numeric values for priority will take precedence. If JobPriorityLow is greater than JobPriorityHigh it indicates that numerically lower values are used to represent higher priority jobs. When both JobPriorityHigh and JobPriorityLow are assigned to zero, the queue does not support priorization.

`AvailableJobSheets : string[]`

Describes the job sheets that are supported by all the printers that this queue can access or the subset of jobsheets that the administrator allows users to select for printers accessible through the owning queue. PrintJobs reference this information by specifying the relevant string values.

2.2.1 Futures

In the future, policies will be added to specify how jobs are removed from the queue (e.g. scheduling policies).

2.3 PrintJob

PrintJob describes the manageable properties and characteristics of a job that is destined for a Printer. The PrintJob does not represent or contain the data stream that describes what should be rendered to the printer but that information can be shown through the use of the PrintJobFile association.

Most of the properties of the PrintJob are a subset of those defined for Printer since each PrintJob needs to show the requirements it has of a Printer if the data stream is to be rendered. The rationale for including such attributes is not because the CIM representation is intended for use by a print subsystem (e.g. the print subsystem uses the information to match jobs to printers) but rather administrators have need for this information:

- A PrintJob may not progress from a PrintQueue to a Printer because a user has specified a set of requirements that cannot be met by any of the available printers.
- Monitoring the types of job being submitted looking for misuse (large jobs, high use of a color printer,etc).

The owner of a PrintJob is described using the Owner property in Job and implementers should make best efforts to ensure that the information is an accurate description of the originating user. However there may be cases where a second queue-to-queue transfer is performed by a system daemon and often such jobs will be identified on the target queue by the print service as being owned by the printer subsystem and not the originating owner.

Print jobs are scoped by the queue (see the association OwningPrintQueue). The rationale for this is that even in a scenario where a print job is moved from one queue to another, this action is often closer to deleting the original print job and submitting a new job for the second queue than simply moving the entire job. The identity of the job as far as the second queue is concerned is, or may be, different to the original identity given to the job. In order to satisfy the need for identifying jobs when they move from queue to queue, the JobOrigination attribute may be assigned a string value that identifies the entity which first created (or submitted) the PrintJob.

PrintJob is derived from Job and adds the properties:

`SystemCreationClassName`

Acts as one of the keys for the Job. This is required because the OwningPrintQueue association makes PrintJob weak to PrintQueue that is derived from JobDestination which in turn is weak to System.

`SystemName`

Acts a one of the keys for the job. See SystemCreationClassName.

`QueueCreationClassName`

Acts as one of the keys for the Job. This is required because the OwningPrintQueue association makes PrintJob weak to PrintQueue that is derived from JobDestination which has CreationClassName as a key.

`QueueName`

Acts a one of the keys for the job. See QueueCreationClassName.

`Jobid: string`

Uniquely identifies the Job on the queue.

`SchedulingInformation : string[]`

Provides further qualifications on when the Job should be submitted to the output device being used by the PrintQueue. An example might specify the job should only be printed at night.

`Jobsize: uint32`

The size of the job (in Kbytes).

`Language:uint16`

The page description language used by the print job.

`MimeTypes:string [ ]`

The mime types used by the job.

`RequiredPaperType: string`

Specifies the paper requirements that this job will have.

`Finishing: uint16 [ ]`

This specifies how the document should be finished and the requirements it has on the printer. The values it provides map onto the Capabilities provided by a printer. For CIM 2.1 these were defined as:

```
{"Unknown", "Other", "Color Printing",
 "Duplex Printing", "Copies", "Collation", "Stapling",
 "Transparency Printing"},
```

“Copies” may be specified in Finishing but an instance of a PrintJob that requires multiple copies being created on the output device should set the Copies property appropriately.

A number of additional Finishings/Capabilities will be added which are discussed in section 2.1.

`Copies : uint32`

Number of copies to print.

`HorizontalResolution : uint32`

Jobs required horizontal resolution in Pixels per Inch.

`VerticalResolution : uint32`

Jobs required vertical resolution in Pixels per Inch.

`Charset : string`

Identifies the character set and encoding method for any textual strings used in the management of this job. Strings specified in this property should conform to the semantics and syntax specified by RFC 2046 and contained in the IANA character-set Registry. Examples include “utf-8”, “iso-8859-1”.

`NaturalLanguage : string`

Identifies the language for any textual strings used in the management of this job. The values for strings contained in these strings are specified by RFC 1766. Examples include “en” for English and “de” for German.

`NumberUp: uint32`

The requested number of print-stream pages that the printer can render on a single side of medium.

`JobStatus : string`

This is inherited from Job but overridden because for PrintJob it is used to provide additional information beyond what its already covered by PrintJobStatus.

`PrintJobStatus : uint16`

Indicates the current status of this job with respect to the queue and printer. Values are:

```
{“Other”, “Unknown”, “Pending”, “Blocked”, ”Complete”, “Printing” }
```

In general jobs will normally be marked as “Pending” which indicates that they will be processed by the PrintService and output onto the media. A job which is marked as “Blocked” will not be processed and a Job marked as “Complete” has been printed. Jobs which are currently being processed by the print service and submitted to either printer or other destination should be marked “Printing”. If “Other” is assigned the JobStatus property should provide an appropriate description.

`TimeCompleted : datetime`

The time at which the job was completed. There is no requirement for a job to remain on the queue once it has been processed but if the job does remain on the queue this property can be set appropriately. For jobs that are not in the “Complete” state this property has no meaning.

`RequiredJobSheets : string[]`

Describes the job sheets that this job requires. Both print queues and printers provide a AvailableJobSheets property that describes the job sheets that are available via that queue or printer.

`JobOriginator : string`

Indicates where the job was originally created. Printer service and protocol implementations that allow the identity of a print job to be tracked as it is passed from entity to entity can use this string to make that identification visible through the model (an example of the information that would be provided here could include the originating user, the queue the job was first submitted to and the name of the system that managed that queue).

2.3.1 Futures

In the future, additional policies will be supported to specify what should happen when a print job completes or when an error condition arises. A decision not to add simple string based actions for the CIM v2.2 model was taken so that no compatibility problems will arise in the future.

2.4 PrintService

The PrintService repeats many of the properties listed for Printer but acts to provide a union of these key attribute values for all of the printers currently (or potentially) accessible through that service. The basis for providing such attributes is informational. A systems administrator or user should be able to locate a print service that provides particular set of needed capabilities without necessarily understanding the relationships between PrintService, PrintQueue and Printer. Furthermore it allows the administrator to effectively limit the visibility of the functionality that may be provided by one or more components that "implement" the overall print service. For example, a printer may provide both PCL and Postscript support but in a particular installation an administrator may wish to specify that only a Postscript printing service is available, without modifying the Printer object to state it does not support PCL.

PrintService is derived from Service and adds the following properties:

`PaperTypesAvailable : string[]`

The types of paper this print service can potentially render a print job on. This is equivalent to the PaperSizesSupported for the Printer object Values include "Letter", "Legal", "NA-10x13-Envelope", "NA-9x12-Envelope", etc. The values given in the strings should conform to the syntax specified by DPA (ISO19175) and the Printer MIB (RFC 1539).

`LanguagesSupported : uint16[]`

An array indicating the print languages natively supported. For example “Mime”, “PCL”, “HPGL”, “PJL”, “PS” etc.

If the “Mime” type is is set then the MimeTypesSupported property should describe which mime types the printer will support. Both mime and regular PDLs may be specified.

`MimeTypesSupported: string []`

If “Mime” is set in LanguagesSupport, then the values in this property specify the mime types that this print service can accept.

`Capabilities: uint16[]`

This specifies the types of finishing that are available for documents submitted to this PrintService. This property generally acts as a union of all the Capabilities properties for the Printer objects that can be accessed through the PrintService. Values include “Color Printing”, “Cover”, “Landscape”, etc.

`CapabilityDescriptions: string []`

An array of strings providing more detailed explanations for any of the Printer features indicated in the Capabilities array. Each entry of this array is related to the entry in the Capabilities array that is located at the same index.

`AvailableFilters: string[]`

Describes the filters that are available within this PrintService. For example if the PrintService has the ability to convert Postscript jobs into PCL then the filter responsible for such conversion could be listed in this property.

When a PrintService is started, the PrintQueues that form part of that service become accessible through the PrintSAP. However, depending on the state of that queue, it may not be possible to submit a job to a printer. A queue must be marked as accepting jobs (QueueAccepting) and, for the output to be printed, marked as enabled (QueueEnabled). This distinction is necessary because starting the print service is equivalent to starting an lp daemon (or equivalent system process) rather than providing more granular control.

The `StartMode` property in Service can be used to specify whether the PrintService should start each time a system is booted.

2.5 PrintSAP

The PrintSAP describes how the print services can be reached and the management for that access.

PrintSAP is derived from SAP and adds the following properties:

`PrintProtocol : uint16[]`

Specifies the print protocols that are supported by this access point.

```
Values {"Other", "SMB", "BSD", "SYSV", "HPNP", "IPP", "LOCAL"}
```

A value of local indicates that the service can only be accessed locally (no networked capability).

`PrintProtocolInfo : string []`

Provides additional information to describe the print protocol that is supported by this access point. In the case of “Other” this string must be used to specify what the protocol is.

In a future version of the model more properties will appear here to provide integration with ProtocolEndpoint.

3. Summary Of Associations

3.1 QueueForPrintService

The QueueForPrintService association allows a PrintService to be associated with a PrintQueue.

The association covers the scenario where a system has multiple print services supporting multiple print queues, and there is a need to describe a mapping between print services and queues. Such scenarios are rare but they do arise, for example a system might support both IPP and Windows printing through distinct services which are independent of one another (e.g. can be started or stopped without effecting other print services).

This is a many-to-many association so that multiple queues can be used by multiple print services and the association need not be instantiated in the situation where a system only provides a single print service.

The property QueueAcceptingFromService indicates whether the print service is able to place jobs on the queue.

3.2 PrinterServicingQueue

The PrinterServicingQueue shows the Printer devices that will be possible destinations for a PrintJob that is placed onto the associated PrintQueue. A single PrintQueue can be serviced by multiple printers. The precise Printer that is selected for a particular PrintJob is implementation specific. There are several scenarios where a Queue may have multiple Printers associated with it. For example:

- A PrintQueue implementation may attempt to load balance across a number of available printers.
- The PrintQueue may select the destination printer on the basis of its capabilities and the PrintJob requirements (e.g. there is a common queue or spooler for a number of printers).

A single printer may be passed jobs from multiple queues. An example of such a situation arises with networked printers or a printer where serial and parallel connectors are to different systems.

The model does not currently attempt to represent how a printer might be selected for a job on a particular queue. A simple action based mechanism would be insufficient because the job may be directed to a printer on the basis of a number of conditions such as job size, job requirements, the owner and even time of day. In the future a policy might be provided for this purpose.

3.3 QueueForwardsToPrintSAP

In general a PrintJob is taken off a PrintQueue and passed to a Printer for processing and output. In transferring the job from the queue to the printer, zero or more filters may be run over the job converting it from one PDL to another or in some other way transcribing the print instructions from one form to another.

On some systems it is possible for a system administrator to configure a print queue that instead of passing jobs to a printer passes jobs to another print queue. This can be represented in the model because it provides useful management information.

However, even in a multi-queued environment it may not always be necessary to use the QueueForwardsToPrintSAP association. It may be possible to roll a number of print queues, as they are managed by the PrintService(s), into a single logical PrintQueue. If there is no need to individually represent the functionality provided by each queue and the print service makes the combination of the queues appear as one, then a single PrintQueue object will be sufficient.

In general every PrintQueue object should be associated with either a Printer (using the PrinterServicingQueue association) or to a PrintSAP (using PrintSAPServicingQueue). A PrintQueue with neither of these associations implies a queue that would be able to accept jobs but has no destination assigned for those jobs.

3.4 HostedJobDestination

This is used to associate a PrintQueue (inheriting from JobDestination) with a System. It identifies the entity where the PrintQueue resides and allows navigation (using HostedAccessPoint) between the PrintSAP, the PrintQueue and its related PrintJobs. It should be noted that the “owning” PrintService for the PrintQueue cannot be identified through the HostedService association as a system may have multiple PrintService instances providing multiple PrintQueue instances. The PrintService should be determined by traversing the QueueForPrintService association which will correctly identify the PrintService that “owns” a PrintQueue.

3.5 OwningPrintQueue

OwningPrintQueue identifies the PrintQueue instance with which a PrintJob is associated. A particular instance of a PrintJob may only reside on a single PrintQueue. A print job only has an identity within its owning queue so this association is used to make PrintJob weak to PrintQueue. The QueuePosition property indicates the location of the job on the queue where a value of one indicates that the job is at the top of the queue and higher integer values indicate jobs further from the head of the queue. A value of zero indicates that the job is complete.

3.6 PrinterServicingJob

The PrinterServicingJob association shows the Printer device that has been assigned to process a particular job.

This association may be instantiated by a PrintService at the time of a job being submitted if the printer can be determined. However it may not be possible to determine the printer until the job has reached the top of the queue and is processed therefore it is not possible to assume this association will always be made. It is, however, recommended that the association is made for any job on the queue that has a PrintJobStatus of “Complete” as this can then be used to determine to which printer the job was sent. A job with PrintJobStatus of “Printing” should also generally indicate which printer is being used, though if the job is being submitted to another print service the association may not be made.

3.7 PrintJobFile

The PrintJobFile association shows the files that are either the source of the print request or hold the spool information associated with a PrintJob.

4. Rationale

This section contains brief explanations that describe some of the discussions and decisions made in developing this model.

4.1 Representing Networked Printers

The term ‘networked printer’ is used within this paper to indicate a printer that is connected directly to the network. This concept is distinguished from a printer that has a direct connection (such as a parallel or serial connection) to a ComputerSystem (where the owning ComputerSystem provides the networked access point for that printer). The decision to differentiate between these two scenarios was taken on the basis that a stand-alone networked printer should be identifiable within the model. Sub-classing from ComputerSystem to produce a new PrinterSystem was considered. But, no specific attributes could be identified for this PrinterSystem subclass. Therefore, a "dedicated" property was added to ComputerSystem to identify a "networked printer".

4.2 Duplication of properties

Many of the properties related to printer capabilities are duplicated in multiple objects. This arises because:

- A Printer needs to state what it can support and what it is currently set to.
- A PrintJob must specify the requirements it has when its data stream is rendered.
- PrintQueues may need to provide further restrictions, for example on the size of job.
- A PrintService may wish to “advertise” certain capabilities.

An attempt was made to provide a single object class that provided such properties and could be referenced by other objects. The object was known as a PrintContext. As well as providing an attribute grouping for printer characteristics, several associations allowed primitive exclusions to be specified, for example an A4 paper setting excludes an A3 paper setting. However, it proved impossible to ensure there was sufficient flexibility in such exclusions (e.g. it may be possible to have any 2 of 3 paper settings). The use of PrintContext was also likely to lead an instance explosion.

Given the Printer object itself has a number of "supports" type properties it seems that this type of information should simply be reflected in the relevant properties (e.g. pdl is the same). This is the same approach as IPP takes where job template attributes specify values for a job, and their equivalent default and supported values are specified in the printer.

4.3 Relationship to system utilities and services

A command such as the UNIX lp is a point of access to a print service and is represented through an appropriate instantiation of a PrintSAP. A service or daemon that provides the functionality that manages a print subsystem is not directly represented in the model as it is the “physical” or real-world entity that implements the PrintService. It is the PrintService that is manageable not necessarily the daemon itself.

4.4 PrintJob weak to PrintQueue

Making PrintJob weak to PrintQueue means that if a job is moved from queue to queue it must be re-instantiated because its keys will change. Implementationally this may be seen as adding overhead but from a modelling perspective it would not be appropriate to give PrintJob an identity at the enterprise level as print jobs are not enterprise objects. This implies that a PrintJob must be weak to something and given a queue acts as a "container" for print jobs it is a useful place to propagate keys from.

Other implementation problems could have arisen if the model had called for PrintJobs to retain their identity as they move from one system or object to another. In a number of real world print implementations, moving a job from one system to another (or even another queue) requires the job to take on a new identity and it may not be possible to track the previous identities of that job. Therefore for a number of the existing print protocols instantiating a PrintJob that has a lifetime identity regardless of its change in state, would be a barrier to using the CIM model unless extensions to the protocols were implement such that the initial PrintJob object could be identified.

For print protocols where such information can be tracked the JobOriginator property can be used to indicate where and how the job was originated.

5. IPP

This section describes the mappings that exist between the CIM model and IPP.

5.1 Printer

printer-uri-supported

Presently a printer is identified within the CIM model by the weak SystemDevice aggregation to System. The system provides sufficient information for identification within the enterprise. This cannot be trivially mapped to a URI which identifies a name space for the interpretation of the location information, but its does ensure a printer can be uniquely identified and represented within the enterprise.

uri-security-supported

See printer-uri-supported

printer-name

Can be mapped in a variety of ways depending on the interpretation of printer-name, e.g. the name of the physical printer, the name of the logical device that provides print functionality or the name that is related to the print functionality that a user actually accesses.

printer-location

Using the PhysicalElementLocation association from the associated PhysicalElement the applicable Location object can be obtained in which Address will specify the printer's location.

printer-info

Maps to Description in ManagedSystemElement.

printer-more-info

Could be inserted into the Description property, inherited from CIM_ManagedSystemElement.

printer-driver-installer

Not mapped.

printer-make-and-model

Maps to Manufacturer, Model and SerialNumber in the associated PhysicalElement.

printer-more-info-manufacturer

Not mapped.

printer-state

Maps to PrinterStatus. An additional state has been added beyond those provided in CIM 2.1 to provide for the stopped state. The additional states that are specified by IPP may require the use of either DetectedErrorState or ErrorInformation to provide additional qualification.

printer-state-reasons

Maps to ErrorInformation.

printer-state-message

Maps to ErrorInformation.

operations-supported

No direct mapping in the Printer object. Many of the operations referred to by operations-supported are provided by the PrintService.

charset-configured

Maps to CurrentCharSet.

charset-supported

Maps to CharSetsSupported.

natural-language-configured

Maps to CurrentNaturalLanguage.

generated-natural-language-supported

Maps to NaturalLanguageSupported.

document-format-default

Maps to DefaultMimeType with DefaultLanguage including “Mime”

document-format-supported

Maps to MimeTypesSupported with LanguagesSupported including “Mime”

printer-is-accepting-jobs

This does not map directly to the CIM Printer object instead the information is available from PrintQueue (see section 5.2)

queued-job-count

This does not map directly to the CIM Printer object instead the information is available from PrintQueue (see section 5.2)

printer-message-from-operator

Not mapped.

color-supported

CIM represents color support by the “Color Printing” value for Capabilities.

reference-uri-schemes-supported

Not currently supported.

pdl-override-supported

Not currently mapped.

printer-up-time

Can be derived using TimeOfLastReset.

printer-current-time

Not directly mapped. From the CIM perspective, a Printer is a device that is a part of a System. Either the printer is directly connected to a ComputerSystem or is aggregated into a "dedicated" ComputerSystem that contains little more than the Printer Device. In either case, the System has an OS and a LocalDateTime property.

multiple-operation-time-out

Not currently mapped.

compression-supported

Not mapped. Currently there are two places where this could be provided. The PrintSAP could advertise that the print service can be accessed using a compressed data stream that describes the overall print job. The Printer could also specify that it accepts a compressed stream between it and its host.

job-k-octets-supported

Maps to MaxSizeSupported.

job-impressions-supported

Not currently mapped.

job-media-sheets-supported

Not currently mapped.

job-sheets-default

Not currently mapped.

multiple-document-handling-default

Not currently mapped.

copies-default

Maps to DefaultCopies.

finishings-default

Maps to selected values of DefaultCapabilities.

sides-default

Maps to selected values of DefaultCapabilities.

number-up-default

Maps to DefaultNumberUp.

orientation-requested-default

Maps to selected values of DefaultCapabilities.

media-default

Maps to DefaultPaperType.

printer-resolution-default

Not currently mapped. HorizontalResolution and VerticalResolution only refer to current values.

print-quality-default

Maps to selected values of DefaultCapabilities.

job-sheets-supported

Not currently supported.

multiple-document-handling-supported

Not currently supported

copies-supported

Maps to MaxCopies.

finishings-supported

Maps to selected values of Capabilities.

page-ranges-supported

Not currently supported.

sides-supported

Maps to selected values of Capabilities.

number-up-supported

Maps to MaxNumberUp.

orientation-requested-supported

Maps to selected values of Capabilities.

media-supported

Maps to selected values of PaperSizesSupported/PaperTypesSupported.

media-ready

Maps to PaperTypesAvailable.

printer-resolution-supported

Not currently supported.

print-quality-supported

Maps to selected values of Capabilities.

5.2 Print Queue

IPP does not provide for a direct representation of a print queue but some of the Printer Description Attributes defined by IPP overlay with the functionality that is modelled with the CIM PrintQueue. In many cases the Printer “abstraction” used by IPP is a union of the PrintQueue and Printer abstractions in the CIM model. This can give rise to information being factored in a different manner because essentially IPP views a Printer and its related services as an atomic object.

printer-name

With the PrintQueue and Printer objects, different names can be assigned to each of these entities. For IPP printer-name is assigned an end-user friendly value to assist in its identification and as noted above this name represents the combined total of both a (possible) queue and the printing device. In general, printer-name is best mapped by the Name property in PrintQueue (inherited from JobDestination) as users will normally access a printer by naming the appropriate print queue. Although the Name property does not uniquely identify a PrintQueue within an enterprise, in many cases it will be used in such a way that it does appear to uniquely identify a queue and related Printer. For example an enterprise may have multiple instances of a PrintQueue with the same name across various systems but to a user on any one of those systems it looks as if there is a single point of access to a particular printer reached through that queue.

printer-is-accepting-jobs

In the CIM model printer-is-accepting-jobs is, in general, equivalent to performing a logical AND between QueueEnabled and QueueAccepting as a job can only reach the printer through the PrintService if both of these properties are TRUE. It should be noted that a networked printer might be unable to distinguish between accepting and the enabling of a queue. As specified by IPP the processing of the queue is independent of the printer’s current state, e.g. the printer may have a paper jam but yet the queue will still have QueueEnabled set as TRUE.

queued-job-count

Equivalent to NumberOnQueue when the PrintQueue is associated with a single Printer Object. If the PrintQueue is supporting multiple printers then it is not possible to determine outstanding jobs for a particular printer. If a printer has a number of supplier queues, it is also not possible to fetch the number of outstanding jobs.

printer-state-reasons

In general, the values of this attribute are only appropriate to the Printer. However spool-area-full is queue related and may be represented by assigning QueueSpoolFull to QueueStatus.

job-k-octets-supported

The maximum size of job could be specified at the printer level, the print service level or the queue level (or any combination). Given a queue may require spool space to store the jobs that are waiting for a particular printer and that an administrator may wish to limit some queues to relatively short jobs, the MaxJobSize property is provided on the PrintQueue object. job-k-octets-supported maps onto this property though a printer may also act to limit the maximum size of a job.

job-impressions-supported

Not mapped.

job-media-sheets-supported

Not mapped.

job-priority-supported

This can be derived from JobPriorityHigh and JobPriorityLow. If both properties are assigned to zero then the PrintQueue does not support prioritization.

job-hold-until-supported

Not currently supported. This may be added as part of the support for policies in a future version of the printer model.

job-priority-default

Maps to JobPriorityDefault.

job-hold-until-default

Not currently supported. This may be added as part of the support for policies in a future version of the printer model.

5.3 Print Job

job-priority

Maps onto priority in PrintJob (which is inherited from Job). The CIM implementation does not limit priorities to the range 0-100 but rather allows the valid range to be expressed on a per queue basis.

job-hold-until

The SchedulingInformation property may be used to provide equivalent descriptions to those specified by IPP such as ‘no-hold’, ‘day-time’ and ‘night’.

job-sheets

In IPP this defines job start and end sheets for the job (none, standard or user defined extras). In the CIM representation this is specified with RequiredJobSheets.

multiple-document-handling

This is only relevant if a job consists of more than one document. It describes how to collate or manage the different documents (single-document, separate-documents-uncollated-copies, separate-documents-collated-copies,single-document-new-sheet). The CIM printer model does not distinguish between the different types of print job but it is possible to describe similar properties for the entire PrintJob (e.g. Finishing, Sides, StackingOrder, etc).

copies

Direct mapping to Copies

finishing

Maps onto Finishing.

page-ranges

Currently not supported in the CIM representation. It is assumed that the actual data associated with a PrintJob is self-contained and representative of the image that should be rendered on the output device.

sides

See Finishing.

number-up

Direct mapping to NumberUp.

orientation-requested

Direct mapping to the orientation values in Finishing.

media

Maps onto RequiredPaperType.

printer-resolution

Maps onto HorizontalResolution and VerticalResolution.

print-quality

Maps onto print quality values in Finishing.

job-uri

No equivalent in CIM.

job-id

Maps to Jobid.

job-printer-uri

No mapping. IPP specifies that this URI should specify which Printer object created the Job but in the CIM representation it is not mandated that a particular object would create the PrintJob as this is an operational behaviour rather than a management description.

job-more-info

Not currently mapped.

job-name

The Name inherited from ManagedSystemElement should be used for this purpose.

job-originating-user-name

Maps to Owner, which is inherited from Job.

job-state

Maps to PrintJobStatus and JobStatus (inherited from Job) which is a string that could carry any of values prescribed by IPP.

job-state-reasons

No mapping at this time. JobStatus could be used to purvey this information.

job-state-message

No mapping at this time.

number-of-documents

None.

output-device-assigned

Maps to the PrinterServicingJob association.

time-at-creation

Maps to TimeSubmitted inherited from Job.

time-at-processing

Not mapped.

time-at-completed

Maps to TimeCompleted, though the CIM representation does not require that finished jobs are instantiated as objects. The persistence of an object for a job need only be the period of time from job creation to the job being taken from the print queue and serviced appropriately.

number-of-intervening-jobs

Can be determined using the QueuePosition property on the OwningPrintQueue association which must be instantiated for every print job.

job-message-from-operator

Not mapped. Such a message would be implemented outside of the CIM representation but the policies that will be support in a future version may provide similar capabilities.

job-k-octets

Maps to JobSize.

job-impressions

Not mapped. A PrintJob may be (re)processed or formatted as part of its general processing as it is taken from the queue and therefore this value cannot be determined ahead of the Printer acting on the job.

job-media-sheets

Not mapped. A PrintJob may be (re)processed or formatted as part of its general processing as it is taken from the queue and therefore this value cannot be determined ahead of the Printer acting on the job.

job-k-octets-processed

Not mapped.

job-impressions-completed

Not mapped.

job-media-sheets-completed

Not mapped.

attributes-charset

Maps to Charset.

attributes-natural-language

Maps to NatualLanguage.

5.4 PrintSAP

operations-supported

For IPP this specifies explicit IPP operations that are available (e.g. Print-Job, Get-Jobs). Some of these operations relate to the printer device and others are queue/print service related. Generically the operations that are available are implied by the protocol. Therefore if additional capability information needs to be portrayed on the PrintSAP, a new class should be derived.

6. IPP Physical Use Case

This section provides an example use case for the physical aspects of a printer. The example shown in Figure 2 is not the only way in which the physical components of a printer could be modelled nor should it be viewed as a recommendation for how such objects should be instantiated. In many situations a modeller may wish to provide other objects to provide management software with more physical information or even create new derived classes that provide further refinement to the information that is specific to printer hardware.

The starting point for this physical representation is the Chassis object that represents the enclosure for the printer hardware. This object is also the point at which a link can be made to the logical model considered elsewhere in this whitepaper. Through the use of the Realizes association, the relationship between the Chassis and the Printer (which is derived from LogicalDevice) can be shown. It is also possible to show how the entire print system is packaged by the using the ComputerSystemPackage association.

The total amount of memory that the hardware is physically able to accept is indicated by creating MemoryCapacity objects which are linked to the Chassis using ElementCapacity. In some cases there may be no such instances (e.g. the printer either has no memory). The amount of memory currently installed into the hardware is indicated by creating instances of PhysicalMemory and using the PackagedComponent aggregation. It should be noted that the model allows the precise configurations of physical memory to be expressed, such as which card in which slot is providing that memory.

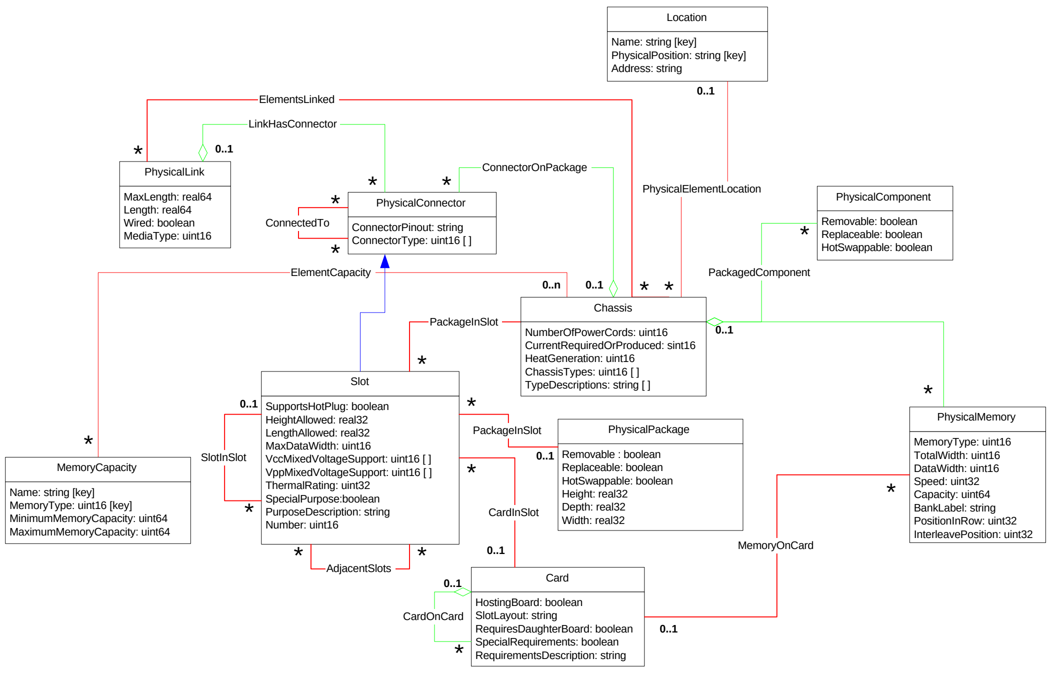

Figure 2. A sample physical use diagram for printers.

A cartridge that can be plugged into the printer could be indicated as an instance of a PhysicalPackage. A cartridge is different to a “simple” card, such as a Network Interface Card (NIC), because it is a standalone physical entity that may aggregate other components such as memory. The association PackageInSlot then shows where the cartridge is plugged in.

Although figure 2 is intended to be an example use, the inheritance of Slot from PhysicalConnector is indicated to demonstrate that the connection type (ConnectorType) used by the slot can be provided through the model. In all likelihood additional information might need to be provided to further qualify the type of connector provided.

Entities such as “Output Trays”, “Console” and “Marker” that are provided within IPP can be represented within the CIM model as either PhysicalPackages (if they consist of multiple components), PhysicalComponents or an object derived from one of these.

Where one coponent needs to be replaced at the same time as another (for example, a drum and toner cartridge) a ReplacementSet object could be provided with the necessary ParticipatesInSet associations.