# 1 Comparison of JDF/1.1 FileSpec and IPP/1.1 and "document-format-details" attributes

From: Tom Hastings
Date: January 22~~1~~, 2003
File: Comparison of JDF and IPP document-format-attrs.doc

This note is a comparison of the JDF/1.1 FileSpec and IPP/1.1 and "document-format-details" attributes. The IPP semantics are taken from [RFC2911] and the IPP Document object specification, version 0.6, 1/13/03. See `ftp://ftp.pwg.org/pub/pwg/fsg/jobticket/IPP_Mapping/ippjdf-mapping-latest.pdf`
The "document-format-detail" attribute is proposed to be added to IPP to meet the requirements of a number of print protocols to have additional information about a Document Format, than just its MIME type. So this attribute has additional fields (member attributes) for version, natural language, platform (OS on which the document was generated), and device ID. It also caters to MIME Media Types that are containers, such as application/zip and multipart/related, where contain additional document of possibly differing MIME media types.

It would be good to add the same attributes to JDF, presumably to the FileSpec resource. Also JDF has a number of attributes which would be good to add to "document-format-details".

JDF issues are highlighted in yellow like this.   IPP issues are highlighted in green like this.

To align these two specs, add 3 attributes to JDF/1.2 FileSpec resource spec: MimeTypeVersion, IEEE1284DeviceId, and DocumentParts and clarify OSVersion..

To align these two specs, add 4 attributes to IPP "document-format-detail" attribute spec: "application-name", "application-name-and-version", "~~platform~~os-name-and-version"~~ (or "os-version")~~, "user-file-name"

Comments and questions on this comparison and the IPP specification are in order.

Here is a comparison of JDF/1.1 FileSpec and IPP/1.1 and IPP "document-format-detail" attribute:

| JDF FileSpec | IPP "document-format-detail" | Comments |
|---|---|---|
| *Application (string)*<br>Creator application, such as Photoshop. | ~~missing~~application-name (name(MAX))<br>The Creator application, such as Photoshop. The version | Add the "application-name" member attribute |

| JDF FileSpec | IPP “document-format-detail” | Comments |
|---|---|---|
| | number MUST NOT be included. See the "application-name-and-version" member attribute. | to IPP "document-format-detail" |
| *AppOS* (enumerations)<br>Operating system of the application that created the file. See *OSVersion* for the version number of the operating system.<br>Possible values are: | ~~ISSUE 05: Should we call this member attribute “os-type”, instead of “platform”, in order to agree with the PWG Printer Installation Extension (see draft-ietf-ipp-install-04.txt)?~~<br>“~~platform~~os-name” (type3 keyword) This ~~REQUIRED~~ member attribute identifies the operating system type ~~the of platform~~ on which the document was generated. Valid values are the special keyword value: ‘unknown’ and the operating system names defined in the IANA document [os-names] that do not include a version number~~and the special keyword value: ‘unknown’~~. Although the IANA registry requires that the names be all upper-case, the values MUST be all lower case in this field since they are IPP keywords (plus hyphen-minus (-), period (.), and slash (/)). Examples: | Same semantics.<br>Clarify JDF *AppOS* |
| *Unknown* – Default value<br>*Mac*<br>*Windows*<br>*Linux*<br>*Solaris*<br>*IRIX*<br>*DG_UX*<br>*HP_UX* | ‘unknown’, ‘linux’, ~~‘linux-2.2’, ‘~~os/2’, ‘mac’, ‘mac-x’, ‘sun-os~~-4.0~~’, ‘unix’, ~~‘unix-bsd’, ‘win32’,~~ ‘windows~~-95~~’, ~~‘windows-98’,~~ ‘windows-ce’, ‘windows-nt’, ~~‘windows-nt-4’, ‘windows-nt-5’,~~ ‘windows-2000’~~, and ‘unknown’.~~ | ~~Could do some better alignment of values and/or add "platform-version" or "os-version"~~ |
| *AppVersion* (string)<br>Version of the value of the *Application* attribute. | application-name-and-version (name(MAX))<br>The application name and version. Example: 'Photoshop V3.0.' | Add "application-name-and-version" to IPP "document-format-detail" |
| *CheckSum* (hexBinary)<br>Checksum of the file being referenced using the RSA MD5 algorithm. In JDF 1.1a, the term RSA MD was completed to RSA MD5. The data type was modified to hexBinary to accommodate the 128 bit output of the MD5 algorithm. | | Handled in IPP by the TLS lower layer security. |

| JDF FileSpec | IPP "document-format-detail" | Comments |
|---|---|---|
| *Compression* (enumeration)<br>Indicates how the file is compressed. Possible values are:<br>*None* – The file is not compressed. Default value.<br>*Deflate* – The file is compressed using ZIP public domain compression (RFC 1951)<br>*Gzip* – GNU zip compression technology (RFC 1952)<br>*Compress* – UNIX compression (RFC 1977) | "compression" (type3 keyword) Values:<br>'none'<br>'deflate'<br>'gzip'<br>'compress' | Is a separate attribute in IPP. Has the exact same values (except for case). |
| *Disposition* (enumeration)<br>Indicates what the device should do with the file when the process that uses this resource as an input resource completes. Possible values are: | | ISSUE: I think LPR has this, right? So should we add it to IPP? |
| *Unlink* – The device should release the file.<br>*Delete* – The device should attempt to delete the file.<br>*Retain* – The device should do nothing with the file. Default value. | | |
| *DocumentNaturalLang* | "document-natural-language" | Same semantics and values |
| *FileFormat* (string)<br>A formatting string used with the *Template* attribute to define a sequence of filenames in a batch process. | | Not sure I understand. Don't put in IPP. |
| *FileSize* (integer) | "k-octets" | Same semantics, different units of measure. |
| *FileTemplate* (string)<br>A template, used with *FileFormat*, to define a sequence of filenames in a batch process. If neither *URL* nor *UID* is present, both *FileFormat* and *FileTemplate* must be present, unless the resource is a pipe. | | Not sure I understand. Don't put in IPP. |
| *FileVersion (string)*<br>Version of the file referenced by this FileSpec. | | Not sure I understand. Don't put in IPP. |
| IEEE1284DeviceId ? (string)<br>Identifies the type of device for which the document was | device-id (text(127))<br>This ~~REQUIRED~~ member attribute identifies the type of | Add IEEE1284DeviceId to JDF FileSpec resource |

| JDF FileSpec | IPP “document-format-detail” | Comments |
|---|---|---|
| formatted, including manufacturer and model.  The value of this variable MUST exactly match the IEEE 1284-2000 Device ID string, except the length field MUST NOT be specified.  See the Microsoft Universal Plug and Play [upnp] section 2.2.6 DeviceId parameter for details and examples. | device for which the document was formatted, including manufacturer and model.  This attribute is intended to identify document formats that are not portable, e.g., PDLs that are device dependent.  The value of this variable MUST exactly match the IEEE 1284-2000 Device ID string, except the length field MUST NOT be specified.  See the Microsoft Universal Plug and Play [upnp] section 2.2.6 DeviceId parameter for details and examples. | |
| *MimeType* (string)<br>Mime type of the file. | “document-format” (mimeMediaType) | Same semantics as JDF, but "document-format" is a separate IPP attribute. |
| MimeTypeVersion ?  (string)<br>The level or version of the document format.  These values are the same as the Printer MIB [RFC1759] prtInterpreterLangLevel.  For example, the value for PostScript level 3 would be: “3” and for PCL 5e would be ‘5e’. | “document-format-version”<br>This ~~REQUIRED~~ member attribute contains the level or version of the document format.   These values are the same as the Printer MIB [RFC1759] prtInterpreterLangLevel.  For example, the value for PostScript level 3 would be: “3” and for PCL 5e would be ‘5e’. | Add MimeTypeVersion to JDF FileSpec resource. |
| *OSVersion* (string)<br>Version of the operating system idnetified by *AppOS*.  Value include 'unknown' and the version number part of the IANA Registry of Operating System Names, not including the HYPHEN (-) character.  See:<br>http://www.iana.org/assignments/operating-system-names<br>Examples:  For *AppOS =* Windows: 95, 98.  For *AppOS =* Windows NT: 4, 5.  For *AppOS = Linux: 2.2.*<br>*ISSUE:  Or should we deprecate OSVersion and define AppOSAndVersion which are complete values from the IANA Registry, which is all uppercase.  Examples are:* ‘LINUX’, ‘LINUX-2.2’, ‘OS/2’, ‘MAC’, ‘MAC-X’, ‘SUN-OS-4.0’, ‘UNIX’, ‘UNIX-BSD’, ‘WIN32’, ‘WINDOWS-95’, ‘WINDOWS-98’, ‘WINDOWS-CE’, ‘WINDOWS-NT’, ‘WINDOWS-NT-4’, ‘WINDOWS-NT-5’, ‘WINDOWS-2000’, AND ‘UNKNOWN’. | os-name-and-version (name(MAX))<br>This member attribute identifies the operating system name and version number on which the document was generated.  Valid values are the special keyword value: ‘unknown’ and any of the operating system names defined in the IANA Registry, but SHOULD be the ones that include a version number.  Although the IANA registry requires that the names be all upper-case, the values MUST be all lower case in this field since they are IPP keywords (plus hyphen-minus (-), period (.), and slash (/)).<br>Examples:  ‘unknown’. ‘linux’, ‘linux-2.2’, ‘os/2’, ‘mac’, ‘mac-x’, ‘sun-os-4.0’, ‘unix’, ‘unix-bsd’, ‘win32’, ‘windows-95’, ‘windows-98’, ‘windows-ce’, ‘windows-nt’, ‘windows-nt-4’, ‘windows-nt-5’, ‘windows-2000’ | Clarify JDF *OSVersion* .<br>Add "os-name-and-version" attribute to IPP "document-format-detail" |

| JDF FileSpec | IPP “document-format-detail” | Comments |
|---|---|---|
| *PageOrder (*enumeration)<br>Indicates whether the pages in the file are in reverse order. Possible values are: | "page-order-received" | separate IPP attribute. Same same semantics |
| *Ascending* – The first page in the file is the lowest numbered page.<br>*Descending* – The first page in the file is the highest numbered page. | '1-to-n'<br>'n-to-1' | Mappable values |
| *ResourceUsage (NMTOKEN)*<br>If an element uses more than one FileSpec subelement, this attribute is used to refer from the parent element to a certain child element of this type, for example, see Error! Reference source not found.. | | Don't put in IPP. |
| *UID (string)*<br>Unique internal ID of the referenced file. This attribute is dependent on the type of file that is referenced:<br>PDF: Variable unique identifier in the ID field of the PDF file’s trailer.<br>ICC Profile: Profile ID in byte 84-99 of the ICC profile header.<br>Others – Format specific. | | Don't put in IPP. |
| *URL (string)*<br>Location of the file. If *URL* is not present, and neither *FileFormat* nor *FileTemplate* are present, the referencing resource must be a pipe. | “document-uri” | Same semantics as JDF, but “document-uri” is a separate IPP attribute. |
| *UserFileName (string)*<br>A user-friendly name which may be used to identify the file, but is not guaranteed to be unique. | user-file-name (name(MAX))<br>A user-friendly name which may be used to identify the file, but is not guaranteed to be unique. | Clarify JDF *UserFileName*<br>Need to add "user-file-name" attribute to IPP "document-format-detail" |
| FileAlias *(element)<br>Defines a set of mappings between file names that may occur in the document and URLs (which may refer to external files or parts of a MIME message). | | Not sure I understand. Don't put in IPP. |
| Add a DocumentPart * subelement which has only a subset | document-format-details (1setOf collection) | ~~Add a DocumentParts~~ |

| JDF FileSpec | IPP “document-format-detail” | Comments |
|---|---|---|
| of the FileSpec attributes, i.e., attributes that can be the same for multiple parts.  So FileSpecPart includes only:<br>*Application (string)*<br>*AppOS*  (enumerations)<br>*AppVersion* (string)<br>*DocumentNaturalLang*<br>*OSVersion* (string)<br>*MimeType* (string)<br>MimeTypeVersion ?  (string)<br>IEEE1284DeviceId ?  (string)<br>Only include the FileSpecPart instances which are different from each other.  So if a .zip file contains 100 PostScript files all with the same attributes, only one DocumentPart subelement is supplied. | This member attribute identifies the document format details of the body parts, if the top level document format is a container types, such as ‘multipart/related’ or ‘application/zip’.  The Printer MUST support this member attribute if it supports a container MIME type.<br>The member attributes defined for this collection are the same as those defined for “document-format-detail” itself, i.e., a recursive definition.  But there MUST NOT be any duplicate collection values at the same level;  its a set, not a sequence.  So 100 PostScript files with the same details in a .zip file would have ‘application/zip’ as the top level MIME type with its details and only one value of the “document-format-details” member attribute with PostScript details. | ~~subelement * to JDF so FileSpec can represent a container objects which contains parts which have their own attributes.~~Add a DocumentPart to JDF FileSpec which has a subset of the FileSpec attributes.  Include only the intensive attributes, i.e., the ones that can apply to multiple parts. ISSUE: Rainer suggest that the sub-parts of a zip file or multipart file can be represented using JDF partitioning, so that we don't need to add sub-elements to FileSpec. |

# 2 Full IPP "document-format-detail" (collection) proposed specification, so far

Here is the full IPP "document-format-detail" specification:

### 2.1.1 document-format-detail (collection)

ISSUE 04: Is the definition of “document-format-detail” OK?

This OPTIONAL Document Description attribute provides additional information about the document format or document formats in the Document object. The member attributes of the “document-format-detail” attribute are:

**Table 1 - "document-format-detail" member attributes**

| Member Attribute Name | Attribute Syntax | Present | Printer Support |
|---|---|---|---|
| document-format | mimeMediaType | MUST | MUST |
| document-format-version | text(127) | MAY | MUST |
| document-format-natural-languages | 1setOf naturalLanguage | MAY | MUST |
| platform | type3 keyword \| name(MAX) | MAY | MUST |
| device-id | text(127) | MAY | MUST |
| document-format-details | 1setOf collection | MAY | MUST - If the Printer support a format that contains other document formats |

#### 2.1.1.1 document-format (mimeMediaType)

This REQUIRED member attribute contains the MIME Media Type of the top level document content. The standard values for this attribute are Internet Media types (sometimes called MIME types). For further details see the description of the 'mimeMediaType' attribute syntax in [RFC2911] section 4.1.9.

If it is a MIME Media Type, such as ‘multipart/related’ or ‘application/zip’, that is a container format that contains document parts, the “document-format-details” member attribute described each part.

#### 2.1.1.2 document-format-version (text(127))

This REQUIRED member attribute contains the level or version of the document format.   For example, the value for PostScript level 3 would be: “3” and for PCL 5e would be ‘5e’.

#### 2.1.1.3 document-format-natural-languages (1setOf naturalLanguage)

This REQUIRED member attribute contains the natural language(s) of the document.  The Printer MAY use this value to select fonts or other Globalization processing.

#### 2.1.1.4 platform (type3 keyword | name(MAX))

ISSUE 05:  Should we call this member attribute “os-type”, instead of “platform”, in order to agree with the PWG Printer Installation Extension (see draft-ietf-ipp-install-04.txt)?

This REQUIRED member attribute identifies the operating system type the of platform on which the document was generated.  Valid values are the operating system names defined in the IANA document [os-names] and the special keyword value: ‘unknown’. Although the IANA registry requires that the names be all upper-case, the values MUST be all lower case in this field since they are IPP keywords (plus hyphen-minus (-), period (.), and slash (/)). Examples: ‘linux’, ‘linux-2.2’, ‘os/2’, ‘mac’, ‘mac-x’, ‘sun-os-4.0’, ‘unix’, ‘unix-bsd’, ‘win32’, ‘windows-95’, ‘windows-98’, ‘windows-ce’, ‘windows-nt’, ‘windows-nt-4’, ‘windows-nt-5’, ‘windows-2000’, and ‘unknown’.

#### 2.1.1.5 device-id (text(127))

This REQUIRED member attribute identifies the type of device for which the document was formatted, including manufacturer and model.  The value of this variable MUST exactly match the IEEE 1284-2000 Device ID string, except the length field MUST not be specified.  See the Microsoft Universal Plug and Play [upnp] section 2.2.6 DeviceId parameter for details and examples.

#### 2.1.1.6 document-format-details (1setOf collection)

This member attribute identifies the document format details of the body parts, if the top level document format is a container types, such as ‘multipart/related’ or ‘application/zip’.  The Printer MUST support this member attribute if it supports a container MIME type.

The member attributes defined for this collection are the same as those defined for “document-format-detail” itself, i.e., a recursive definition.  But there MUST NOT be any duplicate collection values at the same level;  its a set, not a sequence.  So 100 PostScript files with the same details in a .zip file would have ‘application/zip’ as the top level MIME type with its details and only one value of the “document-format-details” member attribute with PostScript details.

### 2.1.2 document-format-detected (mimeMediaType)

This OPTIONAL Document Description attribute is generated by the Printer to indicate the actual document format of the Document object.  This value will differ from the value submitted by the client or defaulted by the Printer, if the Printer automatically senses the document format by examining the document content, such as when the “document-format” attribute has the value: ‘application/octet-stream’ (see [RFC2911] section 4.1.9.1).  Before the document format is known, the Printer populates this attribute with a zero length string.

# 3 Full JDF/1.1 FileSpec specification

Here is the JDF/1.1 FileSpec specification:

## 3.1.1 FileSpec

Specification of a file or a set of files.

### 3.1.1.1.1.1.1.1 Resource Properties

**Resource class:** Parameter
**Resource referenced by:** Error! Reference source not found., Error! Reference source not found., Error! Reference source not found., Error! Reference source not found.
**Example Partition:** *Separation*
**Input of processes:** -
**Output of processes:** -

### 3.1.1.1.1.1.1.2 Resource Structure

| Name | Data Type | Description |
|---|---|---|
| *Application* ? | string | Creator application, such as Photoshop. |
| *AppOS* ? | enumeration | Operating system of the application that created the file. Possible values are:<br>*Unknown* – Default value<br>*Mac*<br>*Windows*<br>*Linux*<br>*Solaris*<br>*IRIX*<br>*DG_UX*<br>*HP_UX* |
| *AppVersion* ? | string | Version of the value of the *Application* attribute. |
| *CheckSum ?*<br>New in JDF 1.1<br>Modified in JDF 1.1A | hexBinary | Checksum of the file being referenced using the RSA MD5 algorithm. In JDF 1.1a, the term RSA MD was completed to RSA MD5. The data type was modified to hexBinary to accommodate the 128 bit output of the MD5 algorithm. |

| Name | Data Type | Description |
|---|---|---|
| *Compression* ? | enumeration | Indicates how the file is compressed. Possible values are:<br>*None* – The file is not compressed. Default value.<br>*Deflate* – The file is compressed using ZIP public domain compression (RFC 1951)<br>*Gzip* – GNU zip compression technology (RFC 1952)<br>*Compress* – UNIX compression (RFC 1977) |
| *Disposition* ? | enumeration | Indicates what the device should do with the file when the process that uses this resource as an input resource completes. Possible values are:<br>*Unlink* – The device should release the file.<br>*Delete* – The device should attempt to delete the file.<br>*Retain* – The device should do nothing with the file. Default value. |
| *DocumentNaturalLang ?* | language | The natural language of the document this FileSpec refers to. |
| *FileFormat* ? | string | A formatting string used with the *Template* attribute to define a sequence of filenames in a batch process.<br>If neither *URL* nor *UID* are present, both *FileFormat* and *FileTemplate* must be present, unless the resource is a pipe. For more information, see the text following this table. |
| *FileSize ?* | integer | Size of the file in Byte. |
| *FileTemplate* ? | string | A template, used with *FileFormat*, to define a sequence of filenames in a batch process. If neither *URL* nor *UID* is present, both *FileFormat* and *FileTemplate* must be present, unless the resource is a pipe. |
| *FileVersion ?*<br>New in JDF 1.1 | string | Version of the file referenced by this FileSpec. |
| *MimeType* ? | string | Mime type of the file. |
| *OSVersion* ? | string | Version of the operating system. |
| *PageOrder ?* | enumeration | Indicates whether the pages in the file are in reverse order. Possible values are:<br>*Ascending* – The first page in the file is the lowest numbered page.<br>*Descending* – The first page in the file is the highest numbered page. |

| Name | Data Type | Description |
|---|---|---|
| *ResourceUsage ?* | NMTOKEN | If an element uses more than one FileSpec subelement, this attribute is used to refer from the parent element to a certain child element of this type, for example, see Error! Reference source not found.. |
| *UID ?*<br>New in JDF 1.1 | string | Unique internal ID of the referenced file. This attribute is dependent on the type of file that is referenced:<br>PDF: Variable unique identifier in the ID field of the PDF file's trailer.<br>ICC Profile: Profile ID in byte 84-99 of the ICC profile header.<br>Others – Format specific. |
| *URL* ? | URL | Location of the file. If *URL* is not present, and neither *FileFormat* nor *FileTemplate* are present, the referencing resource must be a pipe. |
| *UserFileName ?* | string | A user-friendly name which may be used to identify the file. |
| FileAlias * | element | Defines a set of mappings between file names that may occur in the document and URLs (which may refer to external files or parts of a MIME message). |

### *3.1.1.1.1.1.1.3*

### *3.1.1.1.1.1.1.4 Structure of FileAlias Subelement*

| Name | Data Type | Description |
|---|---|---|
| *Alias* | string | The filename which is expected to occur in the file. |
| *Disposition* | enumeration | Indicates what the device should do with the file referenced by this alias when the process that uses this resource as an input resource completes. Possible values are:<br>*Unlink* – The device should release the file.<br>*Delete* – The device should attempt to delete the file.<br>*Retain* – The device should do nothing with the file. |
| *MimeType* ? | string | Mime type of the file. |
| *URL* | URL | The URL which identifies the file the alias refers to. |

### *3.1.1.1.1.1.1.5 Usage of Format and Template*

The function defined when using the attributes *FileFormat* and *FileTemplate* is drawn from the same root as the standard C print function and, therefore, overtly resembles the model of that function. *FileFormat* is the first argument and *FileTemplate* is a comma-separated list of the additional arguments. *FileTemplate* may contain the following operators : +,-,*,/,%,(,) which are evaluated using standard C-operator precedence and the variables defined in the following table:

**Table 3-1 Predefined variables used in FileTemplate**

| Name | Description |
|---|---|
| element | Integer iterator over all elements in a given page. Restarts at 0 for each page. |
| i | Integer iterator over all files produced by this process.  0-based numbering. |
| page | Integer iterator over the page number of a document. This is equivalent to r for the case that each run contains exactly one page. |
| r | Integer iterator over all RunList partitions with a partition key of "*Run*" in an input **RunList**. |
| ri | Integer iterator over all indices in an input Run of a **RunList**. This index is equivalent to looping over a RunIndex. |
| sep | Separation as defined in the separation PartIDKey of a partitioned resource. |
| surf | Surface string, "Front" or "Back" |
| SheetName | SheetName string of a partitioned resource. |
| SignatureName | SignatureName string of a partitioned resource. |
| TileX | X coordinate of a Tile |
| TileY | Y coordinate of a Tile |
| PartVersion | PartVersion string of a partitioned resource. |
| jobPartID | JobPartID string |
| jobID | Job ID string |
| jobName | *DescriptiveName* of the Node that is being processed. |
| Time | Current *Time* in ISO 8601 format. |
| Date | Current *Date* in ISO 8601 format. |
| CustomerID | CustomerID |

Example:

```
<FileSpec FileFormat = "file://here/next/%s/%4.i/m%4.i.pdf" FileTemplate = "JobID,i/100,i%100"/>
```

with JobID = "j001" and a **RunList** defining 2023 created files will iterate all created files and place them into:

```
“file://here/next/j001/0000/m0000.pdf”
…
“file://here/next/j001/0020/m0023.pdf”
```