CIP4 Application Note
Use of the File URL in JDF

Date: 12 November 2003
Version: 0.2
File: App-Note-UseOfTheFileURLInJDF-031112.doc

Table Of Contents

Tables

1 Introduction

The purpose of this appendix is to summarize the valid syntax and recommend best practices of writing file URLs in a JDF file. A file URL is one that starts with the file: URL scheme. The file URL scheme is used to designate files accessible on a particular host computer. This scheme, unlike most other URL schemes, does not designate a resource that is universally accessible over the Internet.

[RFC2396] defines the general syntax for the Uniform Resource Identifier (URI). Furthermore, URIs are either Uniform Resource Locators (URL) (see [RFC1738]) or Uniform Resource Names (URN). A URL locates the object on the network by representing the object's network location, while a URN identifies an object by name without locating it. JDF/1.2 uses URLs exclusively.

Officially, there has been no update to the file URL specification since [RFC 1738]. [RFC 2396] restates a new generic syntax across multiple URL schemes[1], but it does not force schemes to accept any string that matches the generic syntax. [2]

When writing a file URL we should satisfy the specific parts of file URL written in [RFC 1738], and also the generic syntax across multiple schemes from [RFC 2396] as long as it is not in contradiction with [RFC 1738].

2 File URL Generic Syntax

This section summarizes the syntax and semantics of Absolute File URLs and Relative File URLs. See [RFC2396] and [RFC1738]. The RunList/@Directory must have a Base URI, if supplied, which is an Absolute URI. The FileSpec/@URL must have either an Absolute URI or a Relative URI.

2.1 Absolute File URLs

The ABNF for an Absolute File URL is:

fileurl = "file://" [host | "localhost"] "/" fpath [3]

- The file: scheme is NOT case sensitive. (i.e. FILE:// , File:// etc. are also valid).
- The host is a domain name of a network host, or its IP address as a set of decimal digit groups separated by ".". See [RFC2396bis] for IPv6 addresses .
- The host part of the file URL can be the hostname, the host IP address, or the word “localhost”. If omitted, it means localhost and would look like this: file:///fpath.
- Absolute URL is a completely defined URL which includes the scheme, network location (if not local), and all the parts of the URL path (fpath), i.e., an absolute-path reference. The Absolute URL’s fpath must not start with “.” or “..” and must not be relative to any other base URL. In other words, an Absolute URL must have an absolute-path reference.

Examples: File://www.any.com/folder/a.pdf and file:///folder/a.pdf See section 4 for a complete list of examples with explanations.

fpath = fsegment *["/" fsegment]

fsegment = *[uchar | ":" | "@" | "&" | "=" | "+" | "," | "$"] [4]

uchar = unreserved | escape

unreserved = alpha | digit | safe | extra

---

[1] Scheme – The method of access to resources. http://, ftp://, file:// are schemes.

[2] Schemes can ‘overwrite’ the generic syntax with specific definitions. The file: scheme does that. The generic syntax fits any legal file URL but not anything generated from the generic syntax is a legal file URL.

[3] Grammar brief: Rules are separated from definitions by an equal "=", "|" is used to designate alternatives, literals are quoted with "", parentheses "(" and ")" are used to group elements, optional elements are enclosed in "[" and "]" brackets, and elements may be preceded with <n>* to designate n or more repetitions of the following element; n defaults to 0.

[4] The last three characters have been added in [RFC 2396]. See pchar definition in Appendix A. Collected BNF for URI. Page 26.

The “?” character was removed according to the correction in [RFC 1808]. See special note at the end of page 5.

extra = "!" | "*" | "'" | "(" | ")" | "~" [5]

safe = "-" | "_" | "."

alpha = lowalpha | hialpha

lowalpha = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"

hialpha = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"

digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

hex = digit | "A" | "B" | "C" | "D" | "E" | "F" | "a" | "b" | "c" | "d" | "e" | "f"

escape = "%" hex hex

To summarize:

- The following special characters "-" | "_" | "." | "!" | "*" | "'" | "(" | ")" | "~" | "+" | "," | "$" | ":" | "@" | "&" | "=" and the alphanumeric ones: "0"-"9" | "a"-"z" | "A"-"Z" may appear in a file: URL path component (fsegment).
- All other characters should be encoded [6](escaped) if exist.
- Path components should be delimited by the "/" sign.

2.2 Relative File URLs

The ABNF for a Relative URL is:

> Relative-URI = net-path | abs-path | rel-path
> net-path = "//" authority [abs-path]
> abs-path = "/" path-segments
> rel-path = path-segments

Relative File URLs:

- Are defined within [RFC 1808] and [RFC 2396].
- Motivation: Relative addressing of URLs allows document trees to be partially independent of their location and access scheme. For instance, it is possible for a single set of hypertext documents to be simultaneously accessible and traversable via each of the "file", "http", and "ftp" schemes if the documents refer to each other using Relative URLs (since a Relative URL does not have a file scheme specified). Furthermore, document trees can be moved, as a whole, without changing any of the embedded URLs.
- Relative URL are always evaluated with respect to a Base URI (see [RFC2396] and [RFC2396bis]. In JDF, the Base URI is either explicitly supplied as the value of the **RunList***/@Directory* attribute or, if *Directory* is not supplied, as the Absolute URI of the directory containing the JDF file itself.
- Relative URL references are distinguished from Absolute URLs in that Relative URLs do **not** begin with a scheme name. They inherit the scheme part of the Base URI. Relative URLs are one of the following three productions:

  1. a network-path reference (starts with "//") followed by an optional absolute-path. Ex: //www.cip4.org/pub/file.pdf or //www.cip4.org. Note: [RFC2396] says the network-path form is rarely used.

[5] The "~" was added in [RFC 2396]. See Appendix G.2 Modifications from both RFC 1738 and RFC 1808. Page 37.

[6] See the encoding section below. Encode and escape has the same meaning in this context.

Example network-path *with* an absolute path:
Base URL (RunList/@Directory) = “file://machine1/folder1/”
Relative URL (FileSpec/@URL) = “//machine2/folder2/file.pdf”
Resolved URL = “file://machine2/folder2/file.pdf”

Example network-path *without* an absolute path:
Base URL (RunList/@Directory) = “file://machine1/folder1/”
Relative URL (FileSpec/@URL) = “//machine2”
Resolved URL = “file://machine2”

2. an absolute-path reference (starts with "/"). Ex: /folder2/file.pdf

Example:
Base URL (RunList/@Directory) = “file://machine1/folder1/”
Relative URL (FileSpec/@URL) = “/folder2/file.pdf”
Resolved URL = “file://machine1/folder2/file.pdf”

3. a relative-path reference (doesn't start with "/"). Ex: file.pdf, ./file.pdf, ../file.pdf, folder/file.pdf, etc.

- Within a relative-path reference, the complete path segments "." and ".." have special meanings: "the current hierarchy level" and "the level above this hierarchy level", respectively.

Example:
Base URL (RunList/@Directory) = “file://machine1/folder1/”
Relative URL (FileSpec/@URL) = “folder2/file.pdf”
Resolved URL = “file://machine1/folder1/folder2/file.pdf”

Example:
Base URL (RunList/@Directory) = “file://machine1/folder1/”
Relative URL (FileSpec/@URL) = “./folder2/file.pdf”
Resolved URL = “file://machine1/folder1/folder2/file.pdf”

Example:
Base URL (RunList/@Directory) = “file://machine1/folder1/”
Relative URL (FileSpec/@URL) = “../file.pdf”
Resolved URL = “file://machine1/file.pdf”

Example:
Base URL (RunList/@Directory) = “file://machine1/folder1/”
Relative URL (FileSpec/@URL) = “../folder2/file.pdf”
Resolved URL = “file://machine1/folder2/file.pdf”

4. If FileSpec/@URL is a Relative URL, but RunList/@Directory is *not* specified
   then FileSpec/@URL is relative to the container that contains the JDF file itself, i.e., relative to the folder that contains the JDF file itself. The JDF Consumer will know the URL of the folder, in order to process it.

   Example:
   RunList/@Directory *not supplied*
   URL of hot folder: /pub/hot-holder/
   FileSpec/@URL="JDF1.1a-4Sept2002.pdf".
   Resolved URL = /pub/hot-folder/JDF1.1a-4Sept2002.pdf
   Note: This example is well-defined for a UNIX system, but not for Windows or MAC. See use of SystemRoot element in FileFormat/FileTemplate attribute for a system independent way to specify the root.

- JDF XML Example:

```
<RunList ID="FileListLinkOut" Class="Parameter" Status="Available"
Directory="file:///c:/DownloadDir/Title-J626103/">
    <LayoutElement>
      <FileSpec URL="../F-22/job001.pdf"/>
    </LayoutElement>
 </RunList>
```

In the example above the resulting absolute URL would be:
file:///c:/DownloadDir/F-22/job001.pdf
Note: if the trailing "/" in the *Directory* attribute is carelessly omitted, the result would be file:///c:/F-22/job001.pdf, since the URI resolution algorithm removes characters after the right-most "/" character in the Base URI.

3 Escaping of Special Characters in File URLs

File URLs in a JDF must honor the character encoding rules defined in [RFC 2396].

- octets 00-1F and 7F hexadecimal represent control characters; these must be encoded.
- unsafe = space | "<" | ">" | """ | "#" | "%" | "{" | "}" | "|" | "\" | "^" | "[" | "]" | "`"[7]

  All unsafe characters must always be encoded within a URL.
- The characters ";", "/", "?", are the characters, which are **reserved** for special meaning within the file: scheme.
- Usually a URL has the same interpretation when an octet is represented by a character and when it is encoded. However, this is not true for reserved characters:
  Encoding a character reserved for a particular scheme may change the semantics of a URL.
- Characters that are not required to be encoded (including alphanumeric) **may be encoded** within the scheme-specific part of a URL, as long as they are not being used for a reserved purpose.
- Characters may be encoded by a character triplet consisting of the character "%" followed by the two hexadecimal digits (from "0123456789ABCDEF") forming the hexadecimal value of the character. The characters "abcdef" may also be used in hexadecimal encoding.
- A URL must be separated into its path components before the escaped characters within those components can be safely decoded.
- In addition to the character encoding conventions defined in [RFC 2396], all encoded octets must use the encoding declared on the XML document header's encoding attribute.
  For example in case it is <?xml version="1.0" encoding="utf-8"?> - the URL attribute should be UTF-8 encoded.

[7] "~" was removed in [RFC 2396]. See Appendix G.2 Page 37.

To summarize:

- Examples of characters, which **MUST** be encoded in file URL within the path segments:

Table 1 - Characters which MUST be escaped in URL

Char	Escaped char	Char	Escaped char	Char	Escaped char
Space	%20	<	%3C	>	%3E
"	%22	#	%23	%	%25
{	%7B	}	%7D	|	%7C
\	%5C	^	%5E	[	%5B
]	%5D	`	%60	;	%3B
/	%2F	?	%3F		
Control characters	%00-%1F and %7F				
None US-ASCII	UTF-8 encoding examples	Ä	%C3%84	ß	%C3%9F
		é	%C3%A9	力	%E3%82%AB

- In the future, IRI (International Resource Identifier) may allow characters coded above 127 (decimal) directly as well as the escaped form. However, for JDF/1.2 conformance characters above 127 (decimal) must be escaped.
- All other characters may be encoded within the path segments.

It includes the following safe special characters:

"-" | "_" | "." | "!" | "*" | "'" | "(" | ")" | "~" | "+" | "," | "$" | ":" | "@" | "&" | "="

and the alphanumeric ones: "0"-"9" | "a"-"z" | "A"-"Z".

- JDF conforming readers should be capable of reading any escaped file URL even though safe characters were encoded.

4 Examples of file URLs for use in JDF (Informative)

This section show good and bad examples of the file URL and how they are resolved with **RunList**/*@Directory* if supplied. For this example, suppose that RunList/@Directory is "file:///c:/pub/jdf-folder/"

4.1 Good Examples

Table 2 shows good URL examples using the file: URL scheme.

Table 2 - Good URL Examples with the "file:" URL scheme

Example number	Example	Comments
1	<FileSpec URL="File://www.any.com/folder/a.pdf"/>	host is a fully qualified domain name.
2	<FileSpec URL="file://file_server/folder/a.pdf"/>	Can be used to describe UNC path \\file_server\folder\a.pdf
3	<FileSpec URL="File://212.34.55.66/folder/a.pdf"/>	host is an IP address.
4	<FileSpec URL="file://localhost/c:/folder/a.pdf"/>	host is "localhost".
5	<FileSpec URL="FILE:///c%3a/folder/a.pdf"/>	File scheme is case insensitive. /// - host is empty. ":" may be encoded.

Example number	Example	Comments
6	<FileSpec URL="./folder/a.pdf"/>	no scheme - relative path. '.' is the directory specified by RunList/@Directory, if supplied, else '.' is the directory containing the JDF file. This resolves to file:///c:/pub/jdf-folder/folder/a.pdf
7	<FileSpec URL="folder/a.pdf"/>	Same as above, i.e., the leading './' is redundant in a Relative URI.
8	<FileSpec URL="./a.pdf"/>	no scheme - relative path. '.' is the directory specified by RunList/@Directory, if supplied, else '.' is the directory containing the JDF file. So if the JDF file has been dropped into the /pub/hot-folder folder and the RunList/@Directory attribute is omitted, then this Relative URI resolves to: file:///pub/hot-folder/a.pdf
9	<FileSpec URL="a.pdf"/>	Same as above, i.e., the leading './' is redundant in a Relative URI.
10	<FileSpec URL="../../a.pdf"/>	up two levels from "." and take a.pdf. This Relative URI resolves to file:///c:/a.pdf

4.2 Normal examples from [RFC2396] of resolving a URI with a Base URI

Here is the set of Normal examples from section 5.4 of [RFC2396]. To simplify, the query and fragment examples (";p" and "?q" have been removed since they are not used in JDF.

Note: the first token "a" is the authority (host), *not* the first segment in the absolute-path reference and "c" is the current directory (even if there was a path segment to the right). The notes in the right margin have been added by CIP4 and are not in [RFC2396] or [RFC2396bis].

```
5.4 Reference Resolution Examples

   Within an object with a well-defined Base URI (RunList/@Directory)
of:

      http://a/b/c/        -- "//a" is the authority/host name
                           -- "c" is the current directory

   a Relative URI reference would be resolved as follows:
```

Table 3 - Examples of resolved Relative URI references from RFC 2396

Relative URI	Resolved URI	CIP4 Notes
"g:h"	"g:h"	Assumed to be an Absolute URI with URL scheme "g:" (not a device letter). An Absolute URI causes the

Relative URI	Resolved URI	CIP4 Notes
		Base URI to be ignored in resolving.
"g"	"http://a/b/c/g"	“g” is in the current directory “c”
"./g"	"http://a/b/c/g"	same, since “./” is redundant in a Relative URI
"g/"	"http://a/b/c/g/"	The trailing “/” is not significant in the resolved URI (but a trailing “/” is significant in a Base URI)
"/g"	"http://a/g"	Absolute-path replaces the path in the Base URI (a is the authority).
"//g"	"http://g"	The //g authority replaces the “//a” authority
"."	"http://a/b/c/"	The current directory in the Base URI is c, so “.” is c.
"./"	"http://a/b/c/"	same
".."	"http://a/b/"	Up one level from the current directory c, i.e., b.
"../"	"http://a/b/"	same
"../g"	"http://a/b/g"	Up one level to b and then down to g
"../.."	"http://a/"	Up two levels to the root.
"../../"	"http://a/"	same
"../../g"	"http://a/g"	Up two levels to the root then down to g

4.3 Bad Examples

Table 4 lists bad examples of URIs using the "file:" URL scheme.

Table 4 - Bad URI examples using the "file:" URL Scheme

Example number	Example	Comments
21	<FileSpec URL="file:/c:/folder/a.pdf"/>	Host part is missing! Should be file:///c:/… or file://localhost/c:/…
22	<FileSpec URL="file://c:/folder/a.pdf"/>	“c:” is taken as the host name!
23	<FileSpec URL="file:///c:/my docs/a.pdf"/>	space is not a valid character in a URL. Should be: file:///c:/my%20docs/a.pdf
24	<FileSpec URL="file:///c:\folder\a.pdf"/>	“\” is not a valid character in a URL.
25	<FileSpec URL="file:///../a.pdf"/>	It begins with a scheme name and so is an Absolute URI which must not contain ‘.’ or ‘..’. The characters ‘.’ and ‘..’ are characters allowed only in a Relative URI.
26	<FileSpec URL="file:///a.pdf"/>	It has no well-defined meaning, in Windows / Mac OS. What is the exact location of this? In which drive/disk/folder? It has meaning in Unix OS. “/a.pdf”.
27	<FileSpec URL="file://a.pdf "/>	It has no well-defined meaning. Host field is missing !

Example number	Example	Comments
28	<FileSpec URL="file:///c%3a%2ffolder%2fa%2epdf"/>	“/” shouldn’t be escaped ! Should be: file:///c:/folder/a.pdf or file:///c%3a/folder/a%2epdf

4.4 Macintosh Special Care

Mac uses ":" as path separator, but not in the same way as generic URL. In addition path segments can contain the "/" character as well as be "." or "..". It means that we will map like this:

Table 5 - Macintosh Special Cases

Mac	Mapping	URL	Comments
HD1:bar	< = = >	file:///HD1/bar	":" is a path delimiter
HD1:..	< = = >	File:///HD1/%2E%2E	".." is a valid file name in Mac and is encoded.
<undef>	< = =	File:///bar	There is no single root drive for Mac.
HD1:foo/	< = = >	file:///HD1/foo%2F	"/" is a valid file name in Mac and is encoded.
HD1:./foo.txt	< = = >	file:///HD1/.%2Ffoo.txt	"/" is part of the file name and therefore must be encoded.

4.5 Unix Special Care

The Unix file system is easy to map as it uses the same path separator as URLs, has a single root, and segments of "." and ".." have the same meaning.

Table 6 - Unix Special Care

Unix	Mapping	URL	Comments
foo/bar	< = = >	foo/bar	Relative URL.
/foo/bar	< = = >	file:///foo/bar	Absolute URL.
foo:	< = = >	./foo:	":" is a valid file name in Unix.
/	< = = >	File:///	"/" is Unix root directory.
<undef>	< = =	File:///fo%00/bar	%00 is not a legal character

4.6 Semantics of the FileSpec/@URL attribute

The *URL* attribute defines the location of the file. The value of the *URL* attribute must be either:

1. a Relative URI (doesn't start with a URL scheme) which must be one of the following:
    a. net-path – which starts with "//" and may have an abs-path, e.g., //www.cip4.org OR //www.cip4.org/pub/a.pdf
    b. abs-path – which starts with "/" e.g., /pub/a.pdf
    c. rel-path – which does not start with "/" and may contain "." and ".." relative tokens, e.g., a.pdf, folder/a.pdf, ./a.pdf, ../a.pdf, etc.

    See also ##ref Appendix 2.2, Relative File URLs
2. an Absolute URI (starts with a URL scheme), may have an authority/host, and should have an absolute-path reference - e.g., ftp://cip4.org/pub/a.pdf, file:///pub/a.pdf, file:///c:/pub/a.pdf, or cid:pdl.

In the case of a JDF file in a MIME/Multipart/Related package file, the *URL* value may be either a URL or a CID.

If this **FileSpec** also specifies **Container/FileSpec** (meaning that the file associated with this **FileSpec** is contained in a packaging or compression/encoding container), then this *URL* must be a Relative URI and the Base URI is the Absolute URI of where the JDF Consumer extracted the container file.

If this **FileSpec** does *not* specify **Container/FileSpec** (meaning that the file associated with this **FileSpec** is *not* contained in a packaging or compression/encoding container), then this *URL* must be either a Relative URI or an Absolute URI as follows:

(a) If *URL* is a Relative URI, then it is relative to the Base URI. The Base URI is either:

(1) ##refRunList/@Directory, if supplied, else

(2) the URL of the folder containing the JDF file that is being processed (which the accessor has to already know in order to access the JDF).