richtext: Creating Rich Text Strings

D. P. Story
Email: dpstory@acrotex.net

processed July 2, 2020

Contents


```
1 ⟨*package⟩
2 \RequirePackage{xkeyval}
3 \RequirePackage{ifpdf}
4 \RequirePackage{ifxetex}[2006/08/21]
5 \RequirePackage{eforms}
6 \@ifundefined{ifpdfmarkup}{\newif\ifpdfmarkup}{}\pdfmarkupfalse
7 \ifpdf\else\ifxetex\else\pdfmarkuptrue\fi\fi
```

1 Introduction

This package supports the creation of *rich text strings* (a type of pdf string). A rich text string is used in a rich text field as the value of the PDF key **RV**. We also support the **DS** key which determines the default style.

From the PDF Reference (PDF 1.7), page 1310, "these rich text strings are fully-formed XML documents that conform to the rich text conventions specified

for the XML Forms Architecture (XFA) specification, which is itself a subset of the XHTML 1.0 specification, augmented with a restricted set of CSS2 style attributes."

A rich text field may be created using the eforms package, like so

```
\textField[\Ff{\FfRichText}\Ff{\FfMultiline}⟨other-options⟩
  \DS{⟨defaultstyle⟩}\RV{⟨rich-value⟩}\V{⟨plain-value⟩}
]{⟨fldname⟩}{⟨width⟩}{⟨height⟩}
```

This package provides commands and methods for 'conveniently' create values ⟨*rich-value*⟩ and ⟨*defaultstyle*⟩ for **RV** and **DS**; additionally, the value ⟨*plain-value*⟩ of the **V** key is the 'plain' text value of the field; that is the text with all the formatting stripped out.

2 Preliminaries

```
\newif\ifrt@formfield \rt@formfieldtrue
\newif\ifrt@needsbody\rt@needsbodyfalse
\@ifpackageloaded{eforms}%
  {\ifxetex\let\@eqV\@eqnuV\fi}{\rt@needsbodytrue}
\ifxetex\else\hypersetup{pdfencoding=pdfdoc}\fi
\providecommand\eq@RV@Body{<?xml version="1.0"?><body %
  xfa:APIVersion="Acroform:2.7.0.0" %
  xfa:contentType="text/html" %
  xfa:spec="2.1" xmlns="http://www.w3.org/1999/xhtml" %
  xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">}
\providecommand\eq@RV@endBody{</body>}
\def\rt@bBody{\ifrt@needsbody\eq@RV@Body\fi}
\def\rt@eBody{\ifrt@needsbody\eq@RV@endBody\fi}
```

3 Documentation for the **RV** key

We follow the Acrobat user interface. There two tabs of interest Font and paragraph.

Font

Text

- Font: `<font-name>`
- Size: 10
- Baseline Shift: 0 points
- Underline: No Underline, Underline, Double Underline, Word Underline, Word Double Underline
- Style: Bold, Italic, Strike-through
- Color: RGB

Paragraph

Alignment

- Horizontal: left, center, right, justify
- Vertical: top, middle, bottom

Indents

- Left: 0 points
- Right: 0 points
- First: None, First Line, Hanging
- By: 0 points

Spacing

- Above: 0 points
- Below: 0 points
- Line Spacing: Single, 1.5 Lines, Double Lines, Exactly (At: 0 points)

Link Enter a URL for this link

On main Properties Menu bar: Also supported are subscript (`<sub></sub>`) and superscript (`<sup></sup>`)

3.1 The **Font** and **Link** tabs

We support the attributes seen in the above list. We further support subscripts, superscript, and links in this section.

```
\newif\ifrtfontstyle\rtfontstylefalse
```

The keys of the `rtFont` key-value family. Supported keys are `font`, `size`, `raise`, `ulstyle`, `color`, `url`, and `style`. Superscripts and subscripts are handled differently.

`font` The `font` key's value is a font name, if the name contains a space, it should be enclosed in single quotes: `font=Arial` or `font='Myriad Pro'`.

```
\define@key{rtFont}{font}[]{\def\rt@argi{#1}\ifx\rt@argi\@empty
  \let\rf@font\@empty\else\rtfontstyletrue
  \def\rf@font{font-family:#1}\fi}
```

`size` The value of `size` is the size of the font, as measured in points `font=12pt`, note the use of the entity '`pt`'.

```
\define@key{rtFont}{size}[]{\def\rt@argi{#1}\ifx\rt@argi\@empty
  \let\rf@size\@empty\else\rtfontstyletrue\def\rf@size@num{#1}%
  \def\rf@size@pt{#1pt}\def\rf@size{font-size:#1pt}\fi}
\def\rf@size@num{12}\def\rf@size@pt{12pt}
```

`raise` The key `raise` corresponds to the user interface item Baseline Shift, measured in points, for example, `raise=6pt`.

```
\define@key{rtFont}{raise}[]{\def\rt@argi{#1}\ifx\rt@argi\@empty
  \let\rf@raise\@empty\else\rtfontstyletrue
  \def\rf@raise{vertical-align:#1pt}\fi}
```

`ulstyle` The key `ulstyle` supplies an underline attribute, for example, `ul=word` underlines each word in the span.

```
\define@choicekey+{rtFont}{ulstyle}[\val\nr]%
  {none,ul,2ul,wul,2wul}[none]{%
  \ifcase\nr\relax
    \def\rf@ul{none}\or
    \def\rf@ul{underline}\or
    \def\rf@ul{double}\or
    \def\rf@ul{word}\or
    \def\rf@ul{double word}\fi
  \rtfontstyletrue
}{}
```

`color` The key `color` supplies a color attribute for the text in the span. There are two formats: `rrggbb` (hex) and `rgb(r,g,b)` (0-255). For example, `color=ff0000` or `color=rgb(255,0,0)` both color the text red.

```
\def\rt@r{r}\let\rt@One=1 \let\rt@Zero=0
\def\rt@parseColor#1(#2\@nil{\def\rt@argi{#2}\ifx\rt@argi\@empty
\let\rt@rgbdec\rt@Zero\else\let\rt@rgbdec\rt@One\fi}
\def\rt@gobbletonil#1\@nil{}
\bgroup\@makeother\#\@makeother\&%
\gdef\rt@hashtag{#}\gdef\rt@amp{&}\egroup
\define@key{rtFont}{color}[]{\def\rt@argi{#1}\ifx\rt@argi\@empty
  \let\rf@color\@empty\else\rtfontstyletrue
  \rt@parseColor#1(\@nil
  \if\rt@rgbdec\rt@One
    \def\rf@color{color:#1}\else
    \def\rf@color{color:\rt@hashtag#1}\fi
\fi}
\let\rf@color\@empty
```

`url` The key `url` enable the rich text string to contain a URL hypertext link.

```
\define@key{rtFont}{url}[]{\def\rt@argi{#1}\ifx\rt@argi\@empty
  \let\rt@url\@empty\else\rtfontstyletrue\def\rt@url{#1}\fi}
```

`bold` `italic` `strikeit` The `rtFontStyle` family provides keys `bold`, `italic` and `strikeit`. They are possible values of the `style` key, define below. The `style` key can take on one or several values.

```
\define@choicekey+{rtFontStyle}{bold}[\val\nr]{normal,bold}[bold]%
  {\edef\rfS@bold{\val}}{}
\define@choicekey+{rtFontStyle}{italic}[\val\nr]{normal,italic}[italic]%
  {\edef\rfS@italic{\val}}{}
\define@key{rtFontStyle}{strikeit}[]{\def\rfS@strikeit{line-through}}
\let\rfS@normal\@empty\let\rfS@bold\@empty
\let\rfS@italic\@empty\let\rfS@strikeit\@empty
```

`style` Continuing the `rtFont` family, the `style` can take on several values: The key-value of `style={bold,italic,strikeit}` sets the text to bold, italic, and strike out. Multiple attributes must be enclosed in parentheses, as shown above.

```
\define@key{rtFont}{style}[]{\def\rt@argi{#1}\ifx\rt@argi\@empty
  \let\rf@style\@empty\else\rtfontstyletrue\def\rf@style{#1}\fi}
```

raw We provide a `raw` experimental key. The value of this key is passed through; it must be of the proper syntax.

```
\define@key{rtFont}{raw}[]{\def\rt@argi{#1}\ifx\rt@argi\@empty
  \let\rf@raw\@empty\else\rtfontstyletrue\def\rf@raw{#1}\fi}
```

\resetRtFontKeys The is an internal command to reset all keys to their default values.

```
\def\resetRtFontKeys{% rtFont family
  \let\rf@font\@empty\let\rf@size\@empty
  \let\rf@raise\@empty\let\rf@ul\@empty
  \let\rf@color\@empty\let\rf@style\@empty\let\rt@url\@empty
  \let\rf@raw\@empty
% rtFontStyle family
  \let\rfS@normal\@empty\let\rfS@bold\@empty
  \let\rfS@italic\@empty\let\rfS@strikeit\@empty
  \rtfontstylefalse}
```

Now give all keys their default values.

```
\resetRtFontKeys
```

As we interpret the rich text string, we must save it properly formatted in both rich and plain format. These are macro for accumulating the strings.

```
\newcommand{\@AddToRichText}{\g@addto@macro\rt@RichText}
\newcommand{\@AddToPlainText}{\g@addto@macro\rt@PlainText}
```

Some utility commands

```
\def\rt@excl{!}
\def\rt@csarg#1#2{\expandafter#1\csname#2\endcsname}
```

\useRV The `\useRV` command expands to the rich string defined by `\rtpara` and is used as the value of the **RV** key.

\useV The `\useV` command expands to the plain string defined by `\rtpara` and is used as the value of the **V** key.

```
\newcommand{\useRV}[1]{\@nameuse{#1-ri}}
\newcommand{\useV}[1]{\@nameuse{#1-pl}}
```

\rvorvstring `\vorvstring` is similar to `\texorpdfstring`, the first argument is a rich string while the second is a plain string. The two must match correctly, or the PDF reader may not display correctly; on error the reader displays the plain text.

```
\newif\if@rvstring \@rvstringfalse
\def\rvorvstring{\if@rvstring
   \expandafter\@firstoftwo
 \else
   \expandafter\@secondoftwo
 \fi
}
```

\sub
\sup

Subscripts (\sub) and superscripts (\sup) There are two versions, one for rich text expansion and one for plain text expansion. Within \rtpara these two are \let to \sub and \sup.

```
\def\rt@sub#1{\rvorvstring{<sub>#1</sub>}{#1}}
\def\rt@sup#1{\rvorvstring{<sup>#1</sup>}{#1}}
```

\br
\bf
\it

Other supported markup: \br, \bf, and \it.

```
\def\rt@br{\rvorvstring{<br />}{\string\r}}
\def\rt@bf#1{\rvorvstring{<b>#1</b>}{#1}}
\def\rt@it#1{\rvorvstring{<i>#1</i>}{#1}}
\def\rt@spc{\rvorvstring{<span style="xfa-spacerun:yes">\rt@amp
  \rt@hashtag160\rt@SC</span>}{ }}
```

Some convenience commands

```
\def\rt@SC{;}\def\rt@CN{:}\def\rt@fs{font-style}\def\rt@fw{font-weight}
\def\rt@td{text-decoration}
```

The \span command There are two versions of the \span command, these are \rt@remove@span for plain text and \rt@span for rich text strings.

```
\def\rt@remove@span#1#2{#2}
```

\rt@span

This is the internal \span command, it is \let\span\rt@span. Of course \span is a TEX primitive, so we must be careful not to overwrite it.

```
\def\rt@StyleAttr{\ifx\rf@font\@empty\else\rf@font\rt@SC\fi
  \ifx\rf@size\@empty\else\rf@size\rt@SC\fi
  \ifrt@formfield\ifx\rf@raise\@empty\else\rf@raise\rt@SC\fi\fi
  \ifx\rf@ul\@empty\ifx\rfS@strikeit\@empty\else
      \rt@td\rt@CN\rfS@strikeit\rt@SC\fi
  \else\rt@td\rt@CN\rf@ul\ifx\rfS@strikeit\@empty\else\space
      \rfS@strikeit\fi\rt@SC\fi
  \ifx\rfS@bold\@empty\else\rt@fw\rt@CN\rfS@bold\rt@SC\fi
  \ifx\rfS@italic\@empty\else\rt@fs\rt@CN\rfS@italic\rt@SC\fi
  \ifx\rf@color\@empty\else\rf@color\rt@SC\fi
  \ifx\rf@raw\@empty\else\rf@raw\fi}
```

\span

The \span is let to \rt@span within the \rtpara command.

```
\let\rt@afterFont\relax
\newcommand\rt@span[2]{\resetRtFontKeys
  \edef\x{\noexpand\setkeys{rtFont}{#1}}\x\rt@afterFont
  \edef\x{\noexpand\setkeys{rtFontStyle}{\rf@style}}\x
  \edef\rt@Style@ttr{\rt@StyleAttr}%
  \ifx\rt@url\@empty
```

We are processing a regular \span.

```
    \ifx\rt@StyleAttr\@empty\def\x{#2}\else
    \edef\x{<span\ifrtfontstyle\space
    style="\rt@Style@ttr"\fi>#2</span>}\fi
  \else
```

We are processing a \span with the url key set.

```
    \ifx\rt@StyleAttr\@empty\edef\x{<a href="\rt@url">#2</a>}\else
      \edef\x{<a href="\rt@url"
      \ifrtfontstyle style="\rt@Style@ttr">#2</a>\fi}\fi
  \fi
}
```

3.2 The Paragraph tab

We now come to the \rtpara command, which sets the attributes of the Paragraph tab.

\rtpara This is how you define a rich text string, through the use of \rtpara. The command takes three options: (1) The optional first takes key-values just defined in the rtFont and the rtPara families.

```
halign: text-align:left|center|right|justify
valign: text-valign:top|middle|bottom
          top is same as no text-valign attribute vertical-align
Indents:
  Left: margin-left:10pt;
  Right: margin-right:10pt
  First: text-indent: 12pt Indent
         text-indent:-12pt; Hanging
         None
Spacing > Line Spacing Line: height:18pt
Above margin-top:11pt; Below margin-bottom:11pt;
      applies to all text in field, not individual paragraphs
line-height: <num>pt
  Single Space: line-height:\rt@size
  1.5 Lines   : line-height: 1.8*max\rt@size
  Double      : line-height: 2.4*max\rt@size
  Exact       : line-height: <num>pt
```

halign **Alignment** The halign key effects the horizontal alignment of a paragraph, choices are left, center, right, and justify. The default is left.

```
\define@choicekey+{rtPara}{halign}[\val\nr]%
  {left,center,right,justify}[left]{%
  \ifcase\nr\relax
    \def\rt@halign{text-align:left}\or
    \def\rt@halign{text-align:center}\or
    \def\rt@halign{text-align:right}\or
    \def\rt@halign{text-align:justify}\fi
  \rtfontstyletrue
}{}
```

valign The valign seems to effect all paragraphs in the rich text field. Its value deter-

mines the vertical placement of the paragraphs: `top`, `middle`, and `bottom`. The default is `top`.

```
\define@choicekey+{rtPara}{valign}[\val\nr]{top,middle,bottom}[top]{%
  \ifcase\nr\relax
    \def\rt@valign{text-valign:top}\or
    \def\rt@valign{text-valign:middle}\or
    \def\rt@valign{text-valign:bottom}\fi
  \rtfontstyletrue
}{}
\let\rt@halign\@empty\let\rt@valign\@empty
```

margleft
margright

Indents With the `margleft` and `margright` you set the left and right margins of the effected paragraph. The default is `0pt`.

```
\define@key{rtPara}{margleft}{\def\rt@margleft{margin-left:#1pt}}
\define@key{rtPara}{margright}{\def\rt@margright{margin-right:#1pt}}
```

indent
indentby

The `indent` key sets the amount of indent of a paragraph, values are `none`, `first`, and `hanging`. The amount of indent is determined by the key `indentby`, which is set to `12pt` by default.

```
\define@choicekey+{rtPara}{indent}[\val\nr]{none,first,hanging}[none]{%
    \edef\rt@indenttype{\nr}%
}{}
\define@key{rtPara}{indentby}[12]{\def\rt@indentby{#1pt}}
\let\rt@margleft\@empty\let\rt@margright\@empty
\def\rt@indenttype{0}\def\rt@indentby{12pt}
```

margtop
margbottom

Spacing With the `margtop` and `margbottom` you set the space above and below a paragraph. The default is `0pt`.

```
\define@key{rtPara}{margtop}[0]{\def\rt@margtop{#1pt}}
\define@key{rtPara}{margbottom}[0]{\def\rt@margbottom{#1pt}}
\def\rt@margtop{0pt}\def\rt@margbottom{0pt}
```

linespacing

The `linespacing` key the spacing lines. The default is `0pt`. Choices are `single` (spacing), `oneandhalf` (spacing), `double` (spacing), and `exact` (spacing).

```
\define@choicekey{rtPara}{linespacing}[\val\nr]%
  {single,oneandhalf,double,exact}[single]{%line-height
  \edef\rt@linespacingtype{\nr}%
  \ifcase\nr
    \let\rt@linesp\@empty\or
    \setlength{\@tempdima}{1.8pt*\rf@size@num}%
    \edef\rt@linesp{\the\@tempdima}\or
    \setlength{\@tempdima}{2.4pt*\rf@size@num}%
    \edef\rt@linesp{\the\@tempdima}\or
    \def\rt@linesp{\rf@size@pt}\fi
}{}
\let\rt@linesp\@empty\def\rt@linespacingtype{0}
```

lineheight

The `lineheight` key

```
\define@key{rtPara}{lineheight}[]{\def\rt@lineheight{#1pt}}
```

```
\let\rt@lineheight\@empty
```

More convenience definitions.

```
\def\rtp@ti{text-indent}\def\rtp@mt{margin-top}
\def\rtp@mb{margin-bottom}\def\rtp@lh{line-height}
```

We put the parameters all together.

```
\def\rt@ParaAttr{%
% Alignment
  \ifx\rt@halign\@empty\else\rt@halign\rt@SC\fi
\ifrt@formfield
  \ifx\rt@valign\@empty\else\rt@valign\rt@SC\fi
% Indents
  \ifx\rt@margleft\@empty\else\rt@margleft\rt@SC\fi
  \ifx\rt@margright\@empty\else\rt@margright\rt@SC\fi
  \ifcase\rt@indenttype\space\or
    \rtp@ti\rt@CN\rt@indentby\rt@SC\or
    \rtp@ti\rt@CN-\rt@indentby\rt@SC\fi
% Spacing
  \rtp@mt\rt@CN\rt@margtop\rt@SC\rtp@mb\rt@CN\rt@margbottom\rt@SC
  \ifx\rt@linesp\@empty\else
    \if\rt@linespacingtype3%
      \ifx\rt@lineheight\@empty
          \rtp@lh\rt@CN\rt@linesp\rt@SC
      \else
          \rtp@lh\rt@CN\rt@lineheight\rt@SC
      \fi
    \else
        \rtp@lh\rt@CN\rt@linesp\rt@SC
    \fi
  \fi
\fi }
\begingroup
\catcode`\@=0 @catcode`@\=12 @gdef@rtbs{\} @endgroup
\def\rt@cs#1{\rvorvstring{\rtbs\rtbs#1}{\string\134#1}}
```

\rtpara The `\tpara` takes three arguments. The first optional argument is key-value pairs from the `rtFont` and `rtPara` families. The second is a name this rich string. The third is the rich string itself, with supported markup.

```
\let\rt@afterParaFont\relax
```

(2016/10/03) Added the `dir` HTML attribute, default is `"ltr"`.

```
\def\rt@dir{ dir="ltr"}
```

We still have problems with dvips wrapping postscript lines around that break the code. Here we remove all `\spaces`'s with PDF spaces (`\040`), hopefully, there are no spaces at which to break the line and cause harm. `\pdfSP` and `\dl@sp@ce` are defined in insdljs.

```
\def\rt@sp@ce{ }
\def\rtpdfSPDef{\string\040}
\def\rtpdfSPDefPrnt{\string\040\allowbreak}
```

```
\bgroup\obeyspaces
\gdef\makePDFSp{\global\let =\pdfSP}%
\gdef\makeTeXSp{\global\let =\rt@sp@ce}%
\gdef\makeTeXSpPrnt{\global\let =\rtpdfSPDefPrnt}%
\egroup
```

Again, there are problems with EOL that disrupts RV key. Here, we define `\insertPDFSp@tEOL` to insert a PDF space (`\040`) at the end of the line. Seems to work.

```
\bgroup\catcode`\^^M=\active%
\gdef\insertPDFSp@tEOL{%
  \catcode`\^^M=\active%
  \let^^M\rtpdfSPDef%
  \endlinechar=`\^^M}%
\egroup
```

```
\newcommand\rtpara[2][]{\begingroup
  \setkeys{rtPara,rtFont}{#1}\rt@afterParaFont
  \edef\rt@Para@ttr{\rt@ParaAttr\rt@StyleAttr}%
  \global\let\rt@RichText\@empty
  \global\let\rt@PlainText\@empty
  \def\rt@ctrlName{#2}%
```

Before taking the next parameter, we'll make some special definitions.

```
  \def\{{\string\{}\def\}{\string\}}%
  \def\1{\string\1}\def\2{\string\2}\def\3{\string\3}%
  \@makeother\$\@makeother\#\@makeother\^\@makeother\_\@makeother\~%
  \@makeother\&\def\&{\rvorvstring{\string&amp;}{\string&}}% req
  \@makeother\<\def\<{\rvorvstring{\string&lt;}{\string<}}% req
  \@makeother\>\def\>{\rvorvstring{\string&gt;}{\string>}}%
  \@makeother\'\def\'{\rvorvstring{\string&apos;}{\string'}}%
  \@makeother\"\def\"{\rvorvstring{\string&quot;}{\string"}}%
  \let\cs\rt@cs
  \rtpara@cont}
```

`\rtpara` continues with `\rtpara@cont`. For pdfmarkup, we make special definitions that are not needed otherwise. `\makePDFSp` makes the space character into `\040`, `\insertPDFSp@tEOS` insert `\040` at the end of each line.

```
\def\rtpara@cont{\ifpdfmarkup
  \makePDFSp\obeyspaces\insertPDFSp@tEOL\fi
  \@ifnextchar\bgroup\rtpara@cont@i{\expandafter
  \rtpara@cont@i\@gobble}}
```

Now we take the last parameter, which is the rich text markup.

```
\def\rtpara@cont@i#1{%
  \let\sup\rt@sup\let\sub\rt@sub
  \let\br\rt@br\let\bf\rt@bf\let\it\rt@it
  \let\spc\rt@spc
  \let\span\rt@remove@span
  \@rvstringfalse
```

Inserting a space at the beginning of plain text does no harm, but has benefits; it betters matches the plain text with the rich context better.

```
\edef\x{#1}\expandafter\@AddToPlainText\expandafter{\x}%
\let\span\relax
\@rvstringtrue
\rtpara@i#1\span!;\endgroup}
```

Step 1: `\rtpara` comes here.

```
\def\rtpara@i#1\span#2;{\def\argii{#2}%
  \@AddToRichText{#1}%
  \ifx\argii\rt@excl
    \rt@csarg\xdef{\rt@ctrlName-ri}%
    {%
      <p\rt@dir\ifx\rt@Para@ttr\@empty\else\space
        style="\rt@Para@ttr"\fi>\rt@RichText</p>%
     }%
    \rt@csarg\xdef{\rt@ctrlName-pl}{\rt@PlainText}%
    \let\rt@next\relax
  \else
      \def\rt@next{\rtpara@ii#2;}%
  \fi
\rt@next}
```

Step 2: `\rtpara@i` comes here.

```
\def\rtpara@ii#1#2#3;{\def\argii{#2}%
  \ifx\argii\rt@excl
    \let\rt@next\relax
  \else
    \let\span\rt@span
    \let\br\rt@br\let\bf\rt@bf\let\it\rt@it
```

We have encountered `\span{<argi>}{<argii>}` and we expand it appropriately.

```
    \@rvstringtrue
    \span{#1}{#2}\edef\rt@tmp{\noexpand
      \@AddToRichText{\x}}\rt@tmp
    \rt@csarg\xdef{\rt@ctrlName-ri}%
    {%
      <p\rt@dir\ifx\rt@Para@ttr\@empty\else\space
        style="\rt@Para@ttr"\fi>\rt@RichText</p>%
    }%
    \rt@csarg\xdef{\rt@ctrlName-pl}{\rt@PlainText}%
    \let\span\relax
    \def\rt@next{\rtpara@i#3;}%
  \fi\rt@next}
```

`skipline` There is a special definition for `skipline`, `skipline` is used between paragraphs to add a blank line between paragraphs.

```
\rt@csarg\def{par-ri}{}\rt@csarg\def{par-pl}{\string\r}
\rt@csarg\def{skipline-ri}%
  {<p><span style="xfa-spacerun:yes">\rt@amp
    \rt@hashtag160;</span></p>}
```

```
\rt@csarg\def{skipline-pl}{\string\r\space}
\def\rt@skipline{skipline}
\rt@csarg\def{br-ri}{}\rt@csarg\def{br-pl}{\string\r}
```

4 Documentation for the DS key

The **DS** key sets the default style. According to the JavaScript API for Acrobat reference, the default style supports alignment, textFont, (font family, font style, font weight), textColor, and textSize

```
/DSfont: Helvetica,sans-serif 12.0pt;text-align:left;color:#000000
f=this.getField"RichText"
style=f.defaultStyle;
style.fontFamily;
style.fontStyle;
style.fontWeight;
style.textFont;
style.alignment;
style.textColor;
style.textSize;
```

```
\def\rt@DSAttr{\ifx\rf@font\@empty\else\rf@font\rt@SC\fi
  \ifx\rf@size\@empty\else\rf@size\rt@SC\fi
  \ifx\rf@raise\@empty\else\rf@raise\rt@SC\fi
  \ifx\rf@ul\@empty\ifx\rfS@strikeit\@empty\else
    \rt@td\rt@CN\rfS@strikeit\rt@SC\fi
  \else\rt@td\rt@CN\rf@ul\ifx\rfS@strikeit\@empty\else\space
    \rfS@strikeit\fi\rt@SC\fi
  \ifx\rfS@bold\@empty\else\rt@fw\rt@CN\rfS@bold\rt@SC\fi
  \ifx\rfS@italic\@empty\else\rt@fs\rt@CN\rfS@italic\rt@SC\fi
  \ifx\rf@color\@empty\else\rf@color\rt@SC\fi}
```

`\useDefaultDS` A fixed definition for default **DS**.

```
\newcommand\useDefaultDS{font-family:Helvetica,sans-serif;%
  font-size:12.0pt;font-style:normal;font-weight:normal;%
  text-align:left;color:\rt@hashtag000000}
```

`\setDefaultStyle` `\setDefaultStyle{myDS}{font=Arial,...,color=ff0000}`

```
\newcommand{\setDefaultStyle}[2]{\begingroup
  \edef\x{\noexpand\setkeys{rtFont}{#2}}\x
  \edef\x{\noexpand\setkeys{rtFontStyle}{\rf@style}}\x
  \ifx\rf@ul\@empty\else
    \let\rt@ul\@empty\PackageWarning{richtext}{%
      The ul key is not supported within\MessageBreak
      \string\setDefaultStyle. Ignoring it}\fi
  \ifx\rf@raise\@empty\else
    \let\rt@raise\@empty\PackageWarning{richtext}{%
      The raise key is not supported within\MessageBreak
```

```
      \string\setDefaultStyle. Ignoring it}\fi
  \ifx\rt@url\@empty\else\let\rf@url\@empty
    \PackageWarning{richtext}{%
      The url key is not supported within\MessageBreak
      \string\setDefaultStyle. Ignoring it}\fi
```

Fill in any missing essential attributes.

```
  \ifx\rf@font\@empty
    \def\rf@font{font-family:Helvetica,sans-serif}\fi
  \ifx\rf@size\@empty\def\rf@size{font-size:\rf@size@pt}\fi
  \ifx\rf@color\@empty\def\rf@color{color:\rt@hashtag000000}\fi
  \rt@csarg\xdef{#1-DS}{\rt@DSAttr}%
\endgroup}
```

`\useDS` `\useDS{<name>}`

```
\newcommand{\useDS}[1]{\@nameuse{#1-DS}}
```

5 Passing the rich content to RV and V

`\setRVVContent` `\setRVVContent{{name_1}{name_1}...{name_k}}` or `\setRVVContent{name}`

```
\newif\ifrt@firsttok \rt@firsttoktrue
\newif\ifrt@itsskipline \rt@itsskiplinefalse
```

`\rt@addtoRVV` is a convenience internal command to add the name `#1` both to `\toks0`, which holds the **RV** string, and to `\toks2`, which holds the **V** string.

```
\def\rt@addtoRVV#1{%
  \toks4={\useRV{#1}}\edef\rt@tmpRV{\the\toks0\the\toks4}%
  \toks4={\useV{#1}}\edef\rt@tmpV{\the\toks2\the\toks4}%
  \toks0=\expandafter{\rt@tmpRV}\toks2=\expandafter{\rt@tmpV}%
}
```

We begin `\setRVVContent`

```
\def\rt@testifbgroup{\@ifnextchar\bgroup
  {\let\rt@nultiargs=1\rt@gobbletonil}
  {\let\rt@nultiargs=0\rt@gobbletonil}}
\newcommand{\setRVVContent}[2]{\begingroup
  \rt@firsttoktrue \rt@itsskiplinefalse
  \rt@testifbgroup#2\@nil
  \def\contName{#1}\toks0={}\toks2={}\toks4={}%
  \if\rt@nultiargs1\def\rt@next{\setRVVContent@i#2;}\else
  \def\rt@next{\setRVVContent@i{#2};}\fi\rt@next}
\def\setRVVContent@i#1{\def\rt@argi{#1}%
  \ifx\rt@argi\rt@SC
```

If a semi-colon (`\rt@SC`), we are finished. Make the definitions for **RV** and **V**, and exit.

```
    \rt@csarg\xdef{\contName-vcont}{\the\toks0}%
    \rt@csarg\xdef{\contName-pcont}{\the\toks2}%
    \let\rt@next\endgroup
  \else
```

See if the current argument has been declared earlier by `\rtpara`. If not, we issue a warning and ignore it.

```
    \expandafter\ifx\csname #1-ri\endcsname\relax
      \PackageWarning{richtext}
      {The name '#1' is not declared,\MessageBreak
      will ignore it. Check the spelling}%
      \def\rt@next{\setRVVContent@i}%
    \else
```

We want to automatically induce `par` between non-`lineskip` tokens. The automatic `par` goes in prior to the token, so we first skip the first token.

```
      \ifrt@firsttok\rt@firsttokfalse\else
```

Not the first token, see if it is a `skipline`, if yes, register it as a skip line for the next token.

```
        \ifx\rt@argi\rt@skipline
          \rt@itsskiplinetrue
        \else
```

If the previous token was a `skipline`, we don't induce a `par`.

```
          \ifrt@itsskipline
            \rt@addtoRVV{br}\rt@itsskiplinefalse
          \else
```

Finally, if this is not a `skipline`, and the previous token is not a `skipline`, we induce a `par`.

```
            \rt@addtoRVV{par}%
          \fi
        \fi
      \fi
      \rt@addtoRVV{#1}%
      \def\rt@next{\setRVVContent@i}%
    \fi
  \fi\rt@next
}
```

`\useRVContent{`⟨*name*⟩`}` Used to combine several paragraphs (**RV**)

```
\def\useRVContent#1{\@nameuse{#1-vcont}}
```

`\useVContent{`⟨*name*⟩`}` Used to combine several paragraphs (**V**)

```
\def\useVContent#1{\@nameuse{#1-pcont}}
```

6 Typesetting rich text mark up

It may be case that an author may want to display the underlying rich text markup and typeset it into the document for inspection and discussion. For this purpose, we offer

`displayRtPara{`⟨*name*⟩`}` Place an `\rtpara` command within and get a readout of RV and with `\displayRV{`⟨*name*⟩`}` and `\displayV{`⟨*name*⟩`}`.

```
\newenvironment{displayRtPara}[1]{%
  \gdef\displayRtParaName{#1}\let\rtpdfSPDef\rt@sp@ce
  \let\makePDFSp\makeTeXSp\let\rt@spc\rt@sp@ce
  \def\rt@SC{;\allowbreak}\def\rt@CN{:\allowbreak}%
}{%
  \rt@csarg\xdef{displayRV\displayRtParaName}%
    {\useRV{\displayRtParaName}}
  \rt@csarg\xdef{displayV\displayRtParaName}%
    {\useV{\displayRtParaName}}
}
```

`displayRtPara*{`⟨*name*⟩`}` Place an `\rtpara` command within and get a readout of RV and with `\displayRV{`⟨*name*⟩`}` and `\displayV{`⟨*name*⟩`}`. In this version, spaces are displayed as `\040`, designed for the dvips -> distiller workflow. Prints the same result for all other workflows as `displayRtPara`.

```
\newenvironment{displayRtPara*}[1]{%
  \gdef\displayRtParaName{#1}\let\rtpdfSPDef\rtpdfSPDefPrnt
  \let\makePDFSp\makeTeXSpPrnt\let\rt@spc\rtpdfSPDefPrnt
  \def\rt@SC{;\allowbreak}\def\rt@CN{:\allowbreak}%
}{%
  \rt@csarg\xdef{displayRV\displayRtParaName}%
    {\useRV{\displayRtParaName}}
  \rt@csarg\xdef{displayV\displayRtParaName}%
    {\useV{\displayRtParaName}}
}
```

`\displayRV{`⟨*name*⟩`}` displays the **RV** entry as defined by a `\rtpara` command with ⟨*name*⟩. `\displayV` Similarly, `\displayV{`⟨*name*⟩`}` displays the **V** entry

```
\def\displayRV#1{\csname displayRV#1\endcsname}
\def\displayV#1{\csname displayV#1\endcsname}
```

⟨/package⟩

7 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

L

M

O

P

R

S

T

U

V

X

8 Change History

v1.0.4 (2018/08/05)

\rtpara: Inserted space when adding to plain text 10

v1.0.5 (2018/09/25)

\rtpara: Backing off that change 10

v1.0a (2016/09/30)

\setRVVContent: Allow \setRVVContent to have only one argument. 13

v1.0c (2016/10/03)

\rt@span: Added \rt@afterFont to allow access by annot_pro. 6

\rtpara: Added \rt@afterParaFont to allow access by annot_pro. 9

Added \rt@dir 9

v1.1 (2020/06/28)

General: Added \displayRV and \displayV . . . 15

Added displayRtPara and displayRtPara* . 14