adfsymbols

Clea F. Rees*

v1.3 (SVN Rev: 10467) 2024/10/03

Abstract

Hirwen Harendal, Arkandis Digital Foundry (ADF) has produced Symbols ADF. This guide outlines the TEX/LATEX support provided with version 1.001 of the fonts in postscript type 1 format.

1 Introduction

This document explains how to use the TEX/LATEX support included with version 1.001 of the Symbols ADF font collection in postscript type 1 format. The fonts were developed by Hirwen Harendal of the Arkandis Digital Foundry (ADF), and information about the fonts themselves, together with copies of the fonts in opentype format, can be found at http://pagesperso-orange.fr/arkandis/ADF/tugfonts.htm. The fonts are released under the GPL. For details, see README, NOTICE and COPYING.

The TEX/LATEX support package consists of all files listed in manifest.txt and these files are released under the LATEX Project Public Licence as explained in the included licensing notices and README. Please let me know of any problems so that I can solve them if I can. If you can correct the problems and send me the fix, that would be even better. Unlike the fonts themselves, the TEX/LATEX support is somewhat experimental.

adfsymbols includes a copy of the fonts in type 1 format, documentation and support files for TEX/LATEX including two LATEX package files, `adfarrows.sty` and `adfbullets.sty`.

2 The support packages

adfsymbols provides access to the symbols in ArrowsADF and BulletsADF in LATEX through two packages, adfarrows and adfbullets.

2.1 adfarrows

`adfarrows` (*pkg.*) adfarrows provides access to ArrowsADF. The package supports a single option to

*Bug tracker: codeberg.org/cfr/nfssext/issues | Code: codeberg.org/cfr/nfssext | Mirror: github.com/cfr42/nfssext

scale the fonts.

`scale` (*opt.*) = ⟨*scaling factor*⟩

Scale the font by ⟨*scaling factor*⟩, which should be a positive integer or simple decimal such as `2` or `1.2`. This option is intended for cases where the fonts should be scaled to match other fonts used in the document e.g. for consistency with the size of regular text or superscript markers.

Initially empty, which is equivalent to `1` but more efficient.

adfarrows provides the command `\adfarrow{}` which takes a single numerical argument. There are 52 arrows in ArrowsADF which can be produced by feeding the relevant number between 1 and 52 to `\adfarrow{}`.

`\adfarrow` `{`⟨*number*⟩`}`

Where ⟨*number*⟩ is a positive integer between 1 and 52 inclusive[1].

1: ▻	14: ↙	27: ⬸	40: ↙
2: ◅	15: ←	28: ⬻	41: ←
3: ➵	16: ↖	29: ➵	42: ↖
4: ➘	17: ↑	30: ➘	43: ↑
5: ⇓	18: ↗	31: ⇓	44: ↗
6: ➶	19: ⬬	32: ➶	45: ➵
7: ⇐	20: ⬊	33: ⇐	46: ⬊
8: ⬁	21: ⇕	34: ⬊	47: ⇕
9: ⇑	22: ⬈	35: ⇑	48: ⬈
10: ⇗	23: ⬅	36: ⇗	49: ⬸
11: →	24: ⬊	37: →	50: ⬊
12: ↘	25: ⇕	38: ↘	51: ⇑
13: ↓	26: ⬈	39: ↓	52: ⇗

For example, `\adfarrow{5}\adfarrow{9}` produces: ⇓⇑.

2.1.1 Alternative commands

To make things a little more convenient, additional commands are provided to access the various arrows. The effect is to typeset one of the arrows show above but it is not necessary to look up or remember the correct numerical argument.

`\adfhalfarrowright` `\adfhalfarrowleft` `\adfhalfarrowleftsolid` `\adfhalfarrowrightsolid`

First, table 1 lists the four commands provided to access the half arrows. In each case, the number of the arrow is given first. This may be used directly with the `\adfarrow{}` command as explained above. The alternative command is given next. This command may be used to typeset the same arrow. For example both `\adfarrow{1}` and `\adfhalfarrowright` produce ▻. Finally, the arrow produced by the two commands is typeset to their right.

The remaining arrows consist of six families each containing eight arrows — one for each of the eight directions of the compass. These may be accessed in two ways, in addition to using `\adfarrow{}`.

[1]The argument 0 will simply typeset a space and should be avoided as using it may interfere with TEX's spacing algorithms. The problem is that TEX will not recognise it as a space and so will treat it instead as a character.

Table 1: Commands for half arrows

No.	Command		No.	Command	
1	`\adfhalfarrowright`		2	`\adfhalfarrowleft`	
27	`\adfhalfarrowrightsolid`		28	`\adfhalfarrowleftsolid`	

Table 2: Directional commands

Direction	Command	Example usage	
north	`\adfarrown`	`\adfarrown1`	
northeast	`\adfarrowne`	`\adfarrowne2`	
east	`\adfarrowe`	`\adfarrowe3`	
southeast	`\adfarrowse`	`\adfarrowse4`	
south	`\adfarrows`	`\adfarrows5`	
southwest	`\adfarrowsw`	`\adfarrowsw6`	
west	`\adfarroww`	`\adfarroww1`	
northwest	`\adfarrownw`	`\adfarrownw3`	

`\adfarrown` `\adfarrowne` `\adfarrowe` `\adfarrowse` `\adfarrows` `\adfarrowsw` `\adfarroww` `\adfarrownw` `{`⟨*number*⟩`}` First, eight commands are provided (table 2). Each command takes a single numerical argument, ⟨*number*⟩, which must be a positive integer in the range 1–6 inclusive. The argument corresponds to one of the six families of arrows. So using the same number with the different commands will typeset arrows from the same family pointing in different directions.

Second, a further command is provided which allows you to specify both the family and direction as separate arguments. This is in fact the base command `\adfarrow` again. Above, we used the command with just one argument: `\adfarrow{}`. In effect, we left the optional argument empty: `\adfarrow[]{}`.

`\adfarrow` `{`⟨*number*⟩`} [`⟨*family*⟩`]{`⟨*direction*⟩`}`

`\adfarrow`

Where ⟨*number*⟩ may be any positive integer between 1 and 52 (as above), ⟨*family*⟩ may be any integer between 1 and 6 (table 3) and ⟨*direction*⟩ may be any of the eight standard compass directions (table 4). ⟨*family*⟩ may also be the name of the 'family' of arrows. ⟨*direction*⟩ may also be given in an abbreviated form.

When ⟨*family*⟩ is given, the second argument specifies the arrow's direction. *Note that you must specify a family if you specify a direction.* If the optional argument is omitted, the command expects the numerical argument corresponding to the arrow you wish to typeset as listed earlier.

The arrow's direction may be specified in either a long or an abbreviated form.

The different possibilities are illustrated table 5 where each row consists of a selection of equivalent commands which may be used to produce identical output in different ways. In each case, the number of the arrow is given first. This may be used directly with the `\adfarrow{}` command as explained above. One of the eight commands from the previous section follows. Two additional uses of `\adfarrow` are given next using the `\adfarrow[family]{direction}` form described in this section. Finally, the arrow each of these commands typesets is displayed to their right.

Table 3: `\adfarrow`: 'family' names and numbers for first argument

No.	Name
1	opentail
2	plain
3	comic
4	solidtail
5	thick
6	tail

Table 4: `\adfarrow`: direction names for second argument

Direction	Name & abbreviation	
north	north	n
northeast	northeast	ne
east	east	e
southeast	southeast	se
south	south	s
southwest	southwest	sw
west	west	w
northwest	northwest	nw

Table 5: `\adfarrow`: examples

No.	Commands equivalent to `\adfarrow{`⟨*no.*⟩`}`			Result
4	`\adfarrowse1`	`\adfarrow[1]{southeast}`	`\adfarrow[opentail]{se}`	↘
51	`\adfarrown6`	`\adfarrow[tail]{north}`	`\adfarrow[6]{n}`	↘
42	`\adfarrownw5`	`\adfarrow[thick]{nw}`	`\adfarrow[5]{northwest}`	↗
15	`\adfarroww2`	`\adfarrow[2]{w}`	`\adfarrow[plain]{west}`	←
31	`\adfarrows4`	`\adfarrow[solidtail]{south}`	`\adfarrow[4]{s}`	↙
22	`\adfarrowsw3`	`\adfarrow[comic]{sw}`	`\adfarrow[3]{southwest}`	↙

2.2 adfbullets

`adfbullets` (*pkg.*) adfbullets provides access to BulletsADF. The package supports a single option to scale the fonts.

`scale` (*opt.*) = ⟨*scaling factor*⟩

Scale the font by ⟨*scaling factor*⟩, which should be a positive integer or simple decimal such as `2` or `1.2`. This option is intended for cases where the fonts should be scaled to match other fonts used in the document e.g. for consistency with the size of regular text or superscript markers.

Initially empty, which is equivalent to `1` but more efficient.

adfbullets provides the command `\adfbullet{}` which takes a single numerical argument. There are 52 bullets in BulletsADF which can be produced by feeding the relevant number between 1 and 52 to `\adfbullet{}`.

`\adfbullet` `{`⟨*number*⟩`}`

Where ⟨*number*⟩ is a positive integer between 1 and 52 inclusive[2].

1: ✧	14: ✥	27: •	40: ➢
2: ✦	15: ✧	28: •	41: •
3: ✷	16: ✪	29: ■	42: •
4: ✖	17: ✪	30: ◆	43: •
5: ✦	18: ✪	31: ◂	44: ·
6: ✖	19: ✪	32: ▸	45: ◦
7: ✚	20: ✪	33: ▲	46: ▪
8: ✦	21: ✪	34: ▼	47: ▪
9: ✦	22: ✪	35: ◂	48: ✦
10: ✦	23: ✥	36: ▸	49: ✦
11: ✷	24: ✪	37: ◂	50: ◆
12: ✷	25: ✦	38: ▸	51: ✧
13: ✷	26: ✪	39: ◂	52: ◦

For example, `\adfbullet{17}\adfbullet{19}\adfbullet{23}` produces: ✪✪✥.

3 Usage Examples

enumitem allows you to easily change the format of lists:

```
\begin{itemize}[label=\adfbullet{25}]
 \item sealing was,
 \item cabbages;
 \item kings.
\end{itemize}
```

- sealing was,

[2] Again, the argument 0 will simply typeset a space and should be avoided as using it may interfere with TEX's spacing algorithms.

- cabbages;
- kings.

Refer to the package documentation for further details.

adfarrows and adfbullets can be used in beamer presentations to produce lists with custom bullet markers; as icons and markers in pgf diagrams; with sectsty, titlesec and/or fancyhdr to typeset custom headings, headers and footers. For example, the equivalent of,

```
\pagestyle{fancy}
 \fancyhf[ch]{}
 \fancyhf[lf]{}
 \fancyhf[rf]{}
 \fancyhf[lh]{}
 \fancyhf[rh]{}
 \fancyhf[ch]{%
  \itshape adfsymbols\hspace*{1.5em}{\Large\adfbullet{14}}\hspace*{1.5em}\filedate}
 \fancyhf[cf]{%
  \itshape {\large\adfbullet{39}} \thepage~\ofname~\lastpage %
  {\large\adfbullet{40}}}
 \renewcommand{\headrulewidth}{0pt}
```

was used to customise this document's headers and footers with fancyhdr.

A Implementation

You do not need to read the remainder of this document in order to install or use the fonts.

A.1 Encoding

Both ArrowsADF and BulletsADF use a single encoding. The only reason to reencode the fonts is to ensure consecutive slot numbers, which makes the user interface a bit nicer.

```
/SymbolsADFEncoding [
/space
/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
/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
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
```

```
68 /.notdef
69 /.notdef
70 /.notdef
71 /.notdef
72 /.notdef
73 /.notdef
74 /.notdef
75 /.notdef
76 /.notdef
77 /.notdef
78 /.notdef
79 /.notdef
80 /.notdef
81 /.notdef
82 /.notdef
83 /.notdef
84 /.notdef
85 /.notdef
86 /.notdef
87 /.notdef
88 /.notdef
89 /.notdef
90 /.notdef
91 /.notdef
92 /.notdef
93 /.notdef
94 /.notdef
95 /.notdef
96 /.notdef
97 /.notdef
98 /.notdef
99 /.notdef
100 /.notdef
101 /.notdef
102 /.notdef
103 /.notdef
104 /.notdef
105 /.notdef
106 /.notdef
107 /.notdef
108 /.notdef
109 /.notdef
110 /.notdef
111 /.notdef
112 /.notdef
113 /.notdef
114 /.notdef
115 /.notdef
116 /.notdef
117 /.notdef
118 /.notdef
119 /.notdef
120 /.notdef
121 /.notdef
```

122 /.notdef
123 /.notdef
124 /.notdef
125 /.notdef
126 /.notdef
127 /.notdef
128 /.notdef
129 /.notdef
130 /.notdef
131 /.notdef
132 /.notdef
133 /.notdef
134 /.notdef
135 /.notdef
136 /.notdef
137 /.notdef
138 /.notdef
139 /.notdef
140 /.notdef
141 /.notdef
142 /.notdef
143 /.notdef
144 /.notdef
145 /.notdef
146 /.notdef
147 /.notdef
148 /.notdef
149 /.notdef
150 /.notdef
151 /.notdef
152 /.notdef
153 /.notdef
154 /.notdef
155 /.notdef
156 /.notdef
157 /.notdef
158 /.notdef
159 /.notdef
160 /.notdef
161 /.notdef
162 /.notdef
163 /.notdef
164 /.notdef
165 /.notdef
166 /.notdef
167 /.notdef
168 /.notdef
169 /.notdef
170 /.notdef
171 /.notdef
172 /.notdef
173 /.notdef
174 /.notdef
175 /.notdef

176 /.notdef
177 /.notdef
178 /.notdef
179 /.notdef
180 /.notdef
181 /.notdef
182 /.notdef
183 /.notdef
184 /.notdef
185 /.notdef
186 /.notdef
187 /.notdef
188 /.notdef
189 /.notdef
190 /.notdef
191 /.notdef
192 /.notdef
193 /.notdef
194 /.notdef
195 /.notdef
196 /.notdef
197 /.notdef
198 /.notdef
199 /.notdef
200 /.notdef
201 /.notdef
202 /.notdef
203 /.notdef
204 /.notdef
205 /.notdef
206 /.notdef
207 /.notdef
208 /.notdef
209 /.notdef
210 /.notdef
211 /.notdef
212 /.notdef
213 /.notdef
214 /.notdef
215 /.notdef
216 /.notdef
217 /.notdef
218 /.notdef
219 /.notdef
220 /.notdef
221 /.notdef
222 /.notdef
223 /.notdef
224 /.notdef
225 /.notdef
226 /.notdef
227 /.notdef
228 /.notdef
229 /.notdef

```
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
/.notdef
] def
```

adfsymbols: adfarrows

Clea F. Rees*

v1.3 (SVN Rev: 10467) 2024/10/03

```
\NeedsTeXFormat{LaTeX2e}
\RequirePackage{svn-prov}
\ProvidesPackageSVN[\filebase.sty]{$Id: adfarrows.dtx 10467 2024-10-03 23:14:27Z
  cfrees $}[v1.3 \revinfo  ArrowsADF]
\DefineFileInfoSVN[adfarrows]
\newif\if@adfarrows@digonnew
```

Copied verbatim, excepting format and modulo package/module name from Joseph Wright's `siunitx.sty` under LPPL

```
\@ifundefined{ExplLoaderFileDate}{%
  \IfFileExists{expl3.sty}{%
    \RequirePackage{expl3}%
  }{%
    \@adfarrows@digonnewfalse
  }%
}{\@adfarrows@digonnewtrue}
```

`scale` (*opt.*) scale takes a factor by which to scale the fonts. This is empty by default, which is equivalent to `1`, but more efficient.

```
\if@adfarrows@digonnew
\ExplSyntaxOn
\keys_define:nn { adfarrows }
{
  scale .tl_set:N = \adfarrows@scale,
  scale .initial:V = \@empty,
}
\else
  \let\adfarrows@scale\@empty
\fi
```

Provide `\ProcessKeyOptions`, `\IfFormatAtLeastTF` on older kernels. Joseph Wright: from `siunitx.sty` ; https://chat.stackexchange.com/transcript/message/64327823#64327823

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\providecommand \IfFormatAtLeastTF { \@ifl@t@r \fmtversion }
```

*Bug tracker: codeberg.org/cfr/nfssext/issues | Code: codeberg.org/cfr/nfssext | Mirror: github.com/cfr42/nfssext

```
\IfFormatAtLeastTF { 2022-06-01 }
{
  \ProcessKeyOptions [ adfarrows ]
}{
  \RequirePackage { l3keys2e }
  \ProcessKeysOptions { adfarrows }
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\ExplSyntaxOff
```

`\adfarrows@style`

```
\DeclareRobustCommand{\adfarrows@style}{%% do NOT break line below!
  \not@math@alphabet\adfarrows@style\relax
  \fontencoding{U}\fontfamily{ArrowsADF}\fontseries{m}\fontshape{n}\selectfont
}
```

```
\ExplSyntaxOn
```

`\l__adfarrows_base_ot_int`

```
\int_new:N \l__adfarrows_base_ot_int
\int_set:Nn \l__adfarrows_base_ot_int {1}
```

`\l__adfarrows_base_p_int`

```
\int_new:N \l__adfarrows_base_p_int
\int_set:Nn \l__adfarrows_base_p_int {2}
```

`\l__adfarrows_base_c_int`

```
\int_new:N \l__adfarrows_base_c_int
\int_set:Nn \l__adfarrows_base_c_int {3}
```

`\l__adfarrows_base_st_int`

```
\int_new:N \l__adfarrows_base_st_int
\int_set:Nn \l__adfarrows_base_st_int {4}
```

`\l__adfarrows_base_th_int`

```
\int_new:N \l__adfarrows_base_th_int
\int_set:Nn \l__adfarrows_base_th_int {5}
```

`\l__adfarrows_base_t_int`

```
\int_new:N \l__adfarrows_base_t_int
\int_set:Nn \l__adfarrows_base_t_int {6}
```

`\l__adfarrows_dir_e_int`

```
\int_new:N \l__adfarrows_dir_e_int
\int_set:Nn \l__adfarrows_dir_e_int {0}
```

```
\l__adfarrows_dir_east_int
311 \int_new:N \l__adfarrows_dir_east_int
312 \int_set:Nn \l__adfarrows_dir_east_int {0}

\l__adfarrows_dir_se_int
313 \int_new:N \l__adfarrows_dir_se_int
314 \int_set:Nn \l__adfarrows_dir_se_int {1}

l__adfarrows_dir_southeast_int
315 \int_new:N \l__adfarrows_dir_southeast_int
316 \int_set:Nn \l__adfarrows_dir_southeast_int {1}

\l__adfarrows_dir_s_int
317 \int_new:N \l__adfarrows_dir_s_int
318 \int_set:Nn \l__adfarrows_dir_s_int {2}

\l__adfarrows_dir_south_int
319 \int_new:N \l__adfarrows_dir_south_int
320 \int_set:Nn \l__adfarrows_dir_south_int {2}

\l__adfarrows_dir_sw_int
321 \int_new:N \l__adfarrows_dir_sw_int
322 \int_set:Nn \l__adfarrows_dir_sw_int {3}

l__adfarrows_dir_southwest_int
323 \int_new:N \l__adfarrows_dir_southwest_int
324 \int_set:Nn \l__adfarrows_dir_southwest_int {3}

\l__adfarrows_dir_w_int
325 \int_new:N \l__adfarrows_dir_w_int
326 \int_set:Nn \l__adfarrows_dir_w_int {4}

\l__adfarrows_dir_west_int
327 \def\adfarrows@west{west}%
328 \int_new:N \l__adfarrows_dir_west_int
329 \int_set:Nn \l__adfarrows_dir_west_int {4}

\l__adfarrows_dir_nw_int
330 \int_new:N \l__adfarrows_dir_nw_int
331 \int_set:Nn \l__adfarrows_dir_nw_int {5}

l__adfarrows_dir_northwest_int
332 \int_new:N \l__adfarrows_dir_northwest_int
333 \int_set:Nn \l__adfarrows_dir_northwest_int {5}
```

```
\l__adfarrows_dir_n_int
\int_new:N \l__adfarrows_dir_n_int
\int_set:Nn \l__adfarrows_dir_n_int {6}
\l__adfarrows_dir_north_int
\int_new:N \l__adfarrows_dir_north_int
\int_set:Nn \l__adfarrows_dir_north_int {6}
\l__adfarrows_dir_ne_int
\int_new:N \l__adfarrows_dir_ne_int
\int_set:Nn \l__adfarrows_dir_ne_int {7}
l__adfarrows_dir_northeast_int
\int_new:N \l__adfarrows_dir_northeast_int
\int_set:Nn \l__adfarrows_dir_northeast_int {7}
\g__adfarrows_base_int
\int_new:N \g__adfarrows_base_int
\g__adfarrows_add_int
\int_new:N \g__adfarrows_add_int
l__adfarrows_base_opentail_int
\int_new:N \l__adfarrows_base_opentail_int
\int_set:Nn \l__adfarrows_base_opentail_int {3}
\l__adfarrows_base_plain_int
\int_new:N \l__adfarrows_base_plain_int
\int_set:Nn \l__adfarrows_base_plain_int {11}
\l__adfarrows_base_comic_int
\int_new:N \l__adfarrows_base_comic_int
\int_set:Nn \l__adfarrows_base_comic_int {19}
__adfarrows_base_solidtail_int
\int_new:N \l__adfarrows_base_solidtail_int
\int_set:Nn \l__adfarrows_base_solidtail_int {29}
\l__adfarrows_base_thick_int
\int_new:N \l__adfarrows_base_thick_int
\int_set:Nn \l__adfarrows_base_thick_int {37}
```

\l__adfarrows_base_tail_int

```
\int_new:N \l__adfarrows_base_tail_int
\int_set:Nn \l__adfarrows_base_tail_int {45}
```

\l__adfarrows_arrow_int

```
\int_new:N \l__adfarrows_arrow_int
```

__adfarrow_arrow:nn

```
\cs_new_nopar:Nn \__adfarrows_arrow:nn
{
  \int_if_exist:cTF { l__adfarrows_base_#1_int }
  {
    \int_gset_eq:Nc \g__adfarrows_base_int { l__adfarrows_base_#1_int }
  }{ % some kind of error check needed here
    \int_gset:Nn \g__adfarrows_base_int { #1 }
  }
  \int_if_exist:cTF { l__adfarrows_dir_#2_int }
  {
    \int_gset_eq:Nc \g__adfarrows_add_int { l__adfarrows_dir_#2_int }
  }{
    \PackageError{adfarrows}{#2  not a valid direction. Setting east }
    \int_gzero:N \g__adfarrows_add_int
  }
  \int_set:Nn \l__adfarrows_arrow_int { \g__adfarrows_base_int + \g__adfarrows_add_int
  }
  \int_compare:nNnTF { \l__adfarrows_arrow_int } < { 53 }
  { %    \int_compare:nNnTF { \l__adfarrows_arrow_int } > { 0 }
    {
      \expandafter\adfarrows@style\expandafter\char \int_to_arabic:n {
        \l__adfarrows_arrow_int
      }
    }{
      \PackageError{adfarrows}{\textbackslash l__adfarrows_arrow_int must
  be greater than 0 but is \int_to_arabic:n {\l__adfarrows_arrow_int}}%
    }
  }{
    \PackageError{adfarrows}{\textbackslash l__adfarrows_arrow_int   must
  be less than than 53 but is \int_to_arabic:n {\l__adfarrows_arrow_int}}%
  }
}
```

__adfarrow_arrow:n

```
\cs_new_nopar:Nn \__adfarrows_arrow:n
{
  \adfarrows@style\char#1
}
```

\adfarrow

```
\NewDocumentCommand \adfarrow { o m }
{
  \group_begin:
  \IfValueTF { #1 }
  {
    \__adfarrows_arrow:nn { #1 } { #2 }
  }{
    \__adfarrows_arrow:n { #2 }
  }
  \group_end:
}

\ExplSyntaxOff
```

\adfhalfarrowright

```
\newcommand*{\adfhalfarrowright}{\adfarrow{1}}
```

\adfhalfarrowleft

```
\newcommand*{\adfhalfarrowleft}{\adfarrow{2}}
```

\adfhalfarrowrightsolid

```
\newcommand*{\adfhalfarrowrightsolid}{\adfarrow{27}}
```

\adfhalfarrowleftsolid

```
\newcommand*{\adfhalfarrowleftsolid}{\adfarrow{28}}
```

\adfarrowe

```
\gdef\adfarrowe#1{%
 \ifcase #1  \relax
  \or \adfarrow{3}%
  \or \adfarrow{11}%
  \or \adfarrow{19}%
  \or \adfarrow{29}%
  \or \adfarrow{37}%
  \or \adfarrow{45}%
 \fi}
```

\adfarrowse

```
\gdef\adfarrowse#1{%
 \ifcase #1  \relax
  \or \adfarrow{4}%
  \or \adfarrow{12}%
  \or \adfarrow{20}%
  \or \adfarrow{30}%
  \or \adfarrow{38}%
  \or \adfarrow{46}%
 \fi}
```

\adfarrows

```
\gdef\adfarrows#1{%
 \ifcase #1  \relax
  \or \adfarrow{5}%
  \or \adfarrow{13}%
  \or \adfarrow{21}%
  \or \adfarrow{31}%
  \or \adfarrow{39}%
  \or \adfarrow{47}%
 \fi}
```

\adfarrowsw

```
\gdef\adfarrowsw#1{%
 \ifcase #1  \relax
  \or \adfarrow{6}%
  \or \adfarrow{14}%
  \or \adfarrow{22}%
  \or \adfarrow{32}%
  \or \adfarrow{40}%
  \or \adfarrow{48}%
 \fi}
```

\adfarroww

```
\gdef\adfarroww#1{%
 \ifcase #1  \relax
  \or \adfarrow{7}%
  \or \adfarrow{15}%
  \or \adfarrow{23}%
  \or \adfarrow{33}%
  \or \adfarrow{41}%
  \or \adfarrow{49}%
 \fi}
```

\adfarrownw

```
\gdef\adfarrownw#1{%
 \ifcase #1  \relax
  \or \adfarrow{8}%
  \or \adfarrow{16}%
  \or \adfarrow{24}%
  \or \adfarrow{34}%
  \or \adfarrow{42}%
  \or \adfarrow{50}%
 \fi}
```

\adfarrown

```
\gdef\adfarrown#1{%
 \ifcase #1  \relax
  \or \adfarrow{9}%
  \or \adfarrow{17}%
```

```
  \or \adfarrow{25}%
  \or \adfarrow{35}%
  \or \adfarrow{43}%
  \or \adfarrow{51}%
 \fi}
```

`\adfarrowne`

```
\gdef\adfarrowne#1{%
 \ifcase #1  \relax
  \or \adfarrow{10}%
  \or \adfarrow{18}%
  \or \adfarrow{26}%
  \or \adfarrow{36}%
  \or \adfarrow{44}%
  \or \adfarrow{52}%
 \fi}

%% end adfarrows.sty
```

A.2 Font Definitions

`uarrowsadf.fd` (*fd.*) Font declarations for ArrowsADF font

```
\ProvidesFile{uarrowsadf.fd}[v1.3 2024/10/01 font definitions for U/ArrowsADF.]
```

addaswyd o t1phv.fd (dyddiad y ffeil fd: 2020-03-25)

```
  \expandafter\ifx\csname adfarrows@scale\endcsname\relax
    \let\adfarrows@@scale\@empty
  \else
    \edef\adfarrows@@scale{s*[\csname adfarrows@scale\endcsname]}%
  \fi
\DeclareFontFamily{U}{ArrowsADF}{}
\DeclareFontShape{U}{ArrowsADF}{m}{n}{
  <-> \adfarrows@@scale ArrowsADF
}{}
\DeclareFontShape{U}{ArrowsADF}{m}{sc}{<->ssub * ArrowsADF/m/n}{}
\DeclareFontShape{U}{ArrowsADF}{m}{it}{<->ssub * ArrowsADF/m/sc}{}
\DeclareFontShape{U}{ArrowsADF}{m}{sl}{<->ssub * ArrowsADF/m/it}{}
\DeclareFontShape{U}{ArrowsADF}{m}{si}{<->ssub * ArrowsADF/m/sl}{}
\DeclareFontShape{U}{ArrowsADF}{m}{scit}{<->ssub * ArrowsADF/m/si}{}
\DeclareFontShape{U}{ArrowsADF}{m}{scsl}{<->ssub * ArrowsADF/m/scit}{}
\DeclareFontShape{U}{ArrowsADF}{b}{n}{<->ssub * ArrowsADF/m/scsl}{}
\DeclareFontShape{U}{ArrowsADF}{b}{sc}{<->ssub * ArrowsADF/b/n}{}
\DeclareFontShape{U}{ArrowsADF}{b}{it}{<->ssub * ArrowsADF/b/sc}{}
\DeclareFontShape{U}{ArrowsADF}{b}{sl}{<->ssub * ArrowsADF/b/it}{}
\DeclareFontShape{U}{ArrowsADF}{b}{si}{<->ssub * ArrowsADF/b/sl}{}
\DeclareFontShape{U}{ArrowsADF}{b}{scit}{<->ssub * ArrowsADF/b/si}{}
\DeclareFontShape{U}{ArrowsADF}{b}{scsl}{<->ssub * ArrowsADF/b/scit}{}
\DeclareFontShape{U}{ArrowsADF}{bx}{n}{<->ssub * ArrowsADF/b/scsl}{}
\DeclareFontShape{U}{ArrowsADF}{bx}{sc}{<->ssub * ArrowsADF/bx/n}{}
\DeclareFontShape{U}{ArrowsADF}{bx}{it}{<->ssub * ArrowsADF/bx/sc}{}
\DeclareFontShape{U}{ArrowsADF}{bx}{sl}{<->ssub * ArrowsADF/bx/it}{}
```

```
\DeclareFontShape{U}{ArrowsADF}{bx}{si}{<->ssub * ArrowsADF/bx/sl}{}
\DeclareFontShape{U}{ArrowsADF}{bx}{scit}{<->ssub * ArrowsADF/bx/si}{}
\DeclareFontShape{U}{ArrowsADF}{bx}{scsl}{<->ssub * ArrowsADF/bx/scit}{}
```

adfsymbols: adfbullets

Clea F. Rees*

v1.3 (SVN Rev: 10467) 2024/10/03

```
\NeedsTeXFormat{LaTeX2e}
\RequirePackage{svn-prov}
\ProvidesPackageSVN[\filebase.sty]{$Id: adfbullets.dtx 10467 2024-10-03 23:14:27Z
  cfrees $}[v1.3 \revinfo]
\DefineFileInfoSVN[adfbullets]
\newif\if@adfbullets@digonnew
```

Copied verbatim, excepting format and modulo package/module name from Joseph Wright's `siunitx.sty` under LPPL

```
\@ifundefined{ExplLoaderFileDate}{%
  \IfFileExists{expl3.sty}{%
    \RequirePackage{expl3}%
  }{%
    \@adfbullets@digonnewfalse
  }%
}{\@adfbullets@digonnewtrue}
```

`scale` (*opt.*) scale takes a factor by which to scale the fonts. This is empty by default, which is equivalent to `1`, but more efficient.

```
\if@adfbullets@digonnew
\ExplSyntaxOn
\keys_define:nn { adfbullets }
{
  scale .tl_set:N = \adfbullets@scale,
  scale .initial:V = \@empty,
}
\else
  \let\adfbullets@scale\@empty
\fi
```

Provide `\ProcessKeyOptions`, `\IfFormatAtLeastTF` on older kernels. Joseph Wright: from `siunitx.sty` ; https://chat.stackexchange.com/transcript/message/64327823#64327823

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\providecommand \IfFormatAtLeastTF { \@ifl@t@r \fmtversion }
```

*Bug tracker: codeberg.org/cfr/nfssext/issues | Code: codeberg.org/cfr/nfssext | Mirror: github.com/cfr42/nfssext

```
\IfFormatAtLeastTF { 2022-06-01 }
{
  \ProcessKeyOptions [ adfbullets ]
}{
  \RequirePackage { l3keys2e }
  \ProcessKeysOptions { adfbullets }
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\ExplSyntaxOff
```

`\adfbullets@style`

```
\DeclareRobustCommand{\adfbullets@style}{%% do NOT break line below!
  \not@math@alphabet\adfbullets@style\relax
  \fontencoding{U}\fontfamily{BulletsADF}\fontseries{m}\fontshape{n}\selectfont
}
```

`\adfbullet`

```
\newcommand*\adfbullet[1]{{\adfbullets@style\char#1}}
```

```
%% end adfbullets.sty
```

A.3 Font Definitions

`ubulletsadf.fd` (*fd.*) Font declarations for BulletsADF font

```
\ProvidesFile{ubulletsadf.fd}[v1.3 2024/10/01 font definitions for U/BulletsADF.]
```

addaswyd o t1phv.fd (dyddiad y ffeil fd: 2020-03-25)

```
  \expandafter\ifx\csname adfbullets@scale\endcsname\relax
    \let\adfbullets@@scale\@empty
  \else
    \edef\adfbullets@@scale{s*[\csname adfbullets@scale\endcsname]}%
  \fi
\DeclareFontFamily{U}{BulletsADF}{}
\DeclareFontShape{U}{BulletsADF}{m}{n}{
  <-> \adfbullets@@scale BulletsADF
}{}
\DeclareFontShape{U}{BulletsADF}{m}{sc}{<->ssub * BulletsADF/m/n}{}
\DeclareFontShape{U}{BulletsADF}{m}{it}{<->ssub * BulletsADF/m/sc}{}
\DeclareFontShape{U}{BulletsADF}{m}{sl}{<->ssub * BulletsADF/m/it}{}
\DeclareFontShape{U}{BulletsADF}{m}{si}{<->ssub * BulletsADF/m/sl}{}
\DeclareFontShape{U}{BulletsADF}{m}{scit}{<->ssub * BulletsADF/m/si}{}
\DeclareFontShape{U}{BulletsADF}{m}{scsl}{<->ssub * BulletsADF/m/scit}{}
\DeclareFontShape{U}{BulletsADF}{b}{n}{<->ssub * BulletsADF/m/scsl}{}
\DeclareFontShape{U}{BulletsADF}{b}{sc}{<->ssub * BulletsADF/b/n}{}
\DeclareFontShape{U}{BulletsADF}{b}{it}{<->ssub * BulletsADF/b/sc}{}
\DeclareFontShape{U}{BulletsADF}{b}{sl}{<->ssub * BulletsADF/b/it}{}
\DeclareFontShape{U}{BulletsADF}{b}{si}{<->ssub * BulletsADF/b/sl}{}
\DeclareFontShape{U}{BulletsADF}{b}{scit}{<->ssub * BulletsADF/b/si}{}
\DeclareFontShape{U}{BulletsADF}{b}{scsl}{<->ssub * BulletsADF/b/scit}{}
\DeclareFontShape{U}{BulletsADF}{bx}{n}{<->ssub * BulletsADF/b/scsl}{}
```

```
572 \DeclareFontShape{U}{BulletsADF}{bx}{sc}{<->ssub * BulletsADF/bx/n}{}
573 \DeclareFontShape{U}{BulletsADF}{bx}{it}{<->ssub * BulletsADF/bx/sc}{}
574 \DeclareFontShape{U}{BulletsADF}{bx}{sl}{<->ssub * BulletsADF/bx/it}{}
575 \DeclareFontShape{U}{BulletsADF}{bx}{si}{<->ssub * BulletsADF/bx/sl}{}
576 \DeclareFontShape{U}{BulletsADF}{bx}{scit}{<->ssub * BulletsADF/bx/si}{}
577 \DeclareFontShape{U}{BulletsADF}{bx}{scsl}{<->ssub * BulletsADF/bx/scit}{}
```

Change History

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.

Symbols

A