

adfsymbols

Clea F. Rees*

v1.5 (SVN Rev: 11700) 2026/02/25

Abstract

Hirwen Harendal, Arkandis Digital Foundry (ADF) has produced Symbols ADF. This guide outlines the $\text{T}_{\text{E}}\text{X}/\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ support provided with version 1.001 of the fonts in postscript type 1 format.

1 Introduction

This document explains how to use the $\text{T}_{\text{E}}\text{X}/\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 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 $\text{T}_{\text{E}}\text{X}/\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ support package consists of all files listed in manifest.txt and these files are released under the $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 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 $\text{T}_{\text{E}}\text{X}/\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ support is somewhat experimental.

adfsymbols includes a copy of the fonts in type 1 format, documentation and support files for $\text{T}_{\text{E}}\text{X}/\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ including two $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ package files, `adfarrows.sty` and `adfbullets.sty`.

2 The support packages

adfsymbols provides access to the symbols in ArrowsADF and BulletsADF in $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 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 inclusive¹.

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.

`\adhalfarrowright` 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 `\adhalfarrowright` 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{}`.

¹The 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	<code>\adfhalfarrowright</code>	2	<code>\adfhalfarrowleft</code>
27	<code>\adfhalfarrowrightsolid</code>	28	<code>\adfhalfarrowleftsolid</code>

Table 2: Directional commands

Direction	Command	Example usage
north	<code>\adfarrown</code>	<code>\adfarrown1</code> ↑
northeast	<code>\adfarrowne</code>	<code>\adfarrowne2</code> ↗
east	<code>\adfarrowe</code>	<code>\adfarrowe3</code> →
southeast	<code>\adfarrowse</code>	<code>\adfarrowse4</code> ↘
south	<code>\adfarrows</code>	<code>\adfarrows5</code> ↓
southwest	<code>\adfarrowsw</code>	<code>\adfarrowsw6</code> ↙
west	<code>\adfarroww</code>	<code>\adfarroww1</code> ←
northwest	<code>\adfarrownw</code>	<code>\adfarrownw3</code> ↖

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

`\adfarrowsw` Second, a further command is provided which allows you to specify both the family `\adfarroww` 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 `\adfarrownw` effect, we left the optional argument empty: `\adfarrow[]{}.`

`\adfarrow` $\{\langle number \rangle\} [\langle family \rangle] \{\langle direction \rangle\}$

`\adfarrow` Where $\langle number \rangle$ may be any positive integer between 1 and 52 (as above), $\langle family \rangle$ may be any integer between 1 and 6 (table 3) and $\langle direction \rangle$ may be any of the eight standard compass directions (table 4). $\langle family \rangle$ may also be the name of the ‘family’ of arrows. $\langle direction \rangle$ may also be given in an abbreviated form.

When $\langle family \rangle$ 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 <code>\adfarrow{⟨no.⟩}</code>			Result
4	<code>\adfarrowse1</code>	<code>\adfarrow[1]{southeast}</code>	<code>\adfarrow[opentail]{se}</code>	↘
51	<code>\adfarrown6</code>	<code>\adfarrow[tail]{north}</code>	<code>\adfarrow[6]{n}</code>	↖
42	<code>\adfarrownw5</code>	<code>\adfarrow[thick]{nw}</code>	<code>\adfarrow[5]{northwest}</code>	↙
15	<code>\adfarroww2</code>	<code>\adfarrow[2]{w}</code>	<code>\adfarrow[plain]{west}</code>	←
31	<code>\adfarrows4</code>	<code>\adfarrow[solidtail]{south}</code>	<code>\adfarrow[4]{s}</code>	↓
22	<code>\adfarrowsw3</code>	<code>\adfarrow[comic]{sw}</code>	<code>\adfarrow[3]{southwest}</code>	↙

2.2 adfbullets

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

`scale` (*opt.*) = \langle *scaling factor* \rangle

Scale the font by \langle *scaling factor* \rangle , 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` $\{\langle$ *number* $\rangle\}$

Where \langle *number* \rangle is a positive integer between 1 and 52 inclusive².

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,

²Again, 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]{}
\renewcommand{\fancyhf}[ch]{%
  \itshape adfsymbols\hspace*{1.5em}{\Large\adfbullet{14}}\hspace*{1.5em}\filedate}
\renewcommand{\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.

```
1 /SymbolsADFEncoding [
2 /space
3 /A
4 /B
5 /C
6 /D
7 /E
8 /F
9 /G
10 /H
11 /I
12 /J
13 /K
```

14 /L
15 /M
16 /N
17 /O
18 /P
19 /Q
20 /R
21 /S
22 /T
23 /U
24 /V
25 /W
26 /X
27 /Y
28 /Z
29 /a
30 /b
31 /c
32 /d
33 /e
34 /f
35 /g
36 /h
37 /i
38 /j
39 /k
40 /l
41 /m
42 /n
43 /o
44 /p
45 /q
46 /r
47 /s
48 /t
49 /u
50 /v
51 /w
52 /x
53 /y
54 /z
55 /.notdef
56 /.notdef
57 /.notdef
58 /.notdef
59 /.notdef
60 /.notdef
61 /.notdef
62 /.notdef
63 /.notdef
64 /.notdef
65 /.notdef
66 /.notdef
67 /.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

```
230 /.notdef
231 /.notdef
232 /.notdef
233 /.notdef
234 /.notdef
235 /.notdef
236 /.notdef
237 /.notdef
238 /.notdef
239 /.notdef
240 /.notdef
241 /.notdef
242 /.notdef
243 /.notdef
244 /.notdef
245 /.notdef
246 /.notdef
247 /.notdef
248 /.notdef
249 /.notdef
250 /.notdef
251 /.notdef
252 /.notdef
253 /.notdef
254 /.notdef
255 /.notdef
256 /.notdef
257 /.notdef
258 ] def
```

adfsymbols: adfarrows

Clea F. Rees*

v1.5 (SVN Rev: 11700) 2026/02/25

```
259 \NeedsTeXFormat{LaTeX2e}
260 \RequirePackage{svn-prov}
261 \ProvidesPackageSVN[\filebase.sty]{$Id: adfarrows.dtx 11700 2026-02-25 07:11:48Z
  cfrees $}[v1.5 \revinfo ArrowsADF]
262 \DefineFileInfoSVN[adfarrows]
263 \newif\ifadfarrows@digonnew
```

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

```
264 \@ifundefined{ExplLoaderFileDate}{%
265   \IfFileExists{expl3.sty}{%
266     \RequirePackage{expl3}%
267   }{%
268     \@adfarrows@digonnewfalse
269   }%
270 }{\@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.

```
271 \ifadfarrows@digonnew
272 \ExplSyntaxOn
273 \keys_define:mn { adfarrows }
274 {
275   scale .tl_set:N = \adfarrows@scale,
276   scale .initial:V = \@empty,
277 }
278 \else
279 \let\adfarrows@scale\@empty
280 \fi
```

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

```
281 %%%%%%%%%%%
282 \providecommand \IfFormatAtLeastTF { \@ifl@t@r \fmtversion }
```

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

```

283 \IfFormatAtLeastTF { 2022-06-01 }
284 {
285   \ProcessKeyOptions [ adfarrows ]
286 }{
287   \RequirePackage { l3keys2e }
288   \ProcessKeysOptions { adfarrows }
289 }
290 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
291 \ExplSyntaxOff

```

\adfarrows@style

```

292 \DeclareRobustCommand{\adfarrows@style}{%% do NOT break line below!
293   \not@math@alphabet\adfarrows@style\relax
294   \fontencoding{U}\fontfamily{ArrowsADF}\fontseries{m}\fontshape{n}\selectfont
295 }

```

```

296 \RequirePackage{fixtounicode}

```

```

297 \ExplSyntaxOn

```

\l__adfarrows_base_ot_int

```

298 \int_new:N \l__adfarrows_base_ot_int
299 \int_set:Nn \l__adfarrows_base_ot_int {1}

```

\l__adfarrows_base_p_int

```

300 \int_new:N \l__adfarrows_base_p_int
301 \int_set:Nn \l__adfarrows_base_p_int {2}

```

\l__adfarrows_base_c_int

```

302 \int_new:N \l__adfarrows_base_c_int
303 \int_set:Nn \l__adfarrows_base_c_int {3}

```

\l__adfarrows_base_st_int

```

304 \int_new:N \l__adfarrows_base_st_int
305 \int_set:Nn \l__adfarrows_base_st_int {4}

```

\l__adfarrows_base_th_int

```

306 \int_new:N \l__adfarrows_base_th_int
307 \int_set:Nn \l__adfarrows_base_th_int {5}

```

\l__adfarrows_base_t_int

```

308 \int_new:N \l__adfarrows_base_t_int
309 \int_set:Nn \l__adfarrows_base_t_int {6}

```

\l__adfarrows_dir_e_int

```

310 \int_new:N \l__adfarrows_dir_e_int
311 \int_set:Nn \l__adfarrows_dir_e_int {0}

```

\l__adfarrows_dir_east_int

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

\l__adfarrows_dir_se_int

```
314 \int_new:N \l__adfarrows_dir_se_int
315 \int_set:Nn \l__adfarrows_dir_se_int {1}
```

\l__adfarrows_dir_southeast_int

```
316 \int_new:N \l__adfarrows_dir_southeast_int
317 \int_set:Nn \l__adfarrows_dir_southeast_int {1}
```

\l__adfarrows_dir_s_int

```
318 \int_new:N \l__adfarrows_dir_s_int
319 \int_set:Nn \l__adfarrows_dir_s_int {2}
```

\l__adfarrows_dir_south_int

```
320 \int_new:N \l__adfarrows_dir_south_int
321 \int_set:Nn \l__adfarrows_dir_south_int {2}
```

\l__adfarrows_dir_sw_int

```
322 \int_new:N \l__adfarrows_dir_sw_int
323 \int_set:Nn \l__adfarrows_dir_sw_int {3}
```

\l__adfarrows_dir_southwest_int

```
324 \int_new:N \l__adfarrows_dir_southwest_int
325 \int_set:Nn \l__adfarrows_dir_southwest_int {3}
```

\l__adfarrows_dir_w_int

```
326 \int_new:N \l__adfarrows_dir_w_int
327 \int_set:Nn \l__adfarrows_dir_w_int {4}
```

\l__adfarrows_dir_west_int

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

\l__adfarrows_dir_nw_int

```
331 \int_new:N \l__adfarrows_dir_nw_int
332 \int_set:Nn \l__adfarrows_dir_nw_int {5}
```

\l__adfarrows_dir_northwest_int

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

\l__adfarrows_dir_n_int

335 \int_new:N \l__adfarrows_dir_n_int
336 \int_set:Nn \l__adfarrows_dir_n_int {6}

\l__adfarrows_dir_north_int

337 \int_new:N \l__adfarrows_dir_north_int
338 \int_set:Nn \l__adfarrows_dir_north_int {6}

\l__adfarrows_dir_ne_int

339 \int_new:N \l__adfarrows_dir_ne_int
340 \int_set:Nn \l__adfarrows_dir_ne_int {7}

\l__adfarrows_dir_northeast_int

341 \int_new:N \l__adfarrows_dir_northeast_int
342 \int_set:Nn \l__adfarrows_dir_northeast_int {7}

\g__adfarrows_base_int

343 \int_new:N \g__adfarrows_base_int

\g__adfarrows_add_int

344 \int_new:N \g__adfarrows_add_int

\l__adfarrows_base_opentail_int

345 \int_new:N \l__adfarrows_base_opentail_int
346 \int_set:Nn \l__adfarrows_base_opentail_int {3}

\l__adfarrows_base_plain_int

347 \int_new:N \l__adfarrows_base_plain_int
348 \int_set:Nn \l__adfarrows_base_plain_int {11}

\l__adfarrows_base_comic_int

349 \int_new:N \l__adfarrows_base_comic_int
350 \int_set:Nn \l__adfarrows_base_comic_int {19}

\l__adfarrows_base_solidtail_int

351 \int_new:N \l__adfarrows_base_solidtail_int
352 \int_set:Nn \l__adfarrows_base_solidtail_int {29}

\l__adfarrows_base_thick_int

353 \int_new:N \l__adfarrows_base_thick_int
354 \int_set:Nn \l__adfarrows_base_thick_int {37}

`\l__adfarrows_base_tail_int`

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

`\l__adfarrows_arrow_int`

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

I don't know why somebody would use these fonts with a Unicode engine, but, just in case, map for that as well as pdfTeX.

```
358 \bool_if:nT { \sys_if_engine luatex_p: || \sys_if_engine pdftex_p: }
359 {
```

`__adfarrows_glyphtounicode_seq` This seems ... insane?

It would be more efficient to just set everything directly, but this is easier to set up and only read once. First, a sequence to hold glyph names.

```
360 \seq_new:N \l__adfarrows_glyphtounicode_seq
361 \seq_set_from_clist:Nn \l__adfarrows_glyphtounicode_seq
362 {
```

outlines

```
363 A, %% A right arrow top half 21C0
364 B, %% B left arrow top half 21BC
```

outline shaft/tail with solid tip

```
365 C, %% C → 2192
366 D, %% D ↘ 2198
367 E, %% E ↓ 2193
368 F, %% F ↙ 2199
369 G, %% G ← 2190
370 H, %% H ↖ 2196
371 I, %% I ↑ 2191
372 J, %% J ↗ 2197
```

solid in various styles

```
373 K, %% K → 2192
374 L, %% L ↘ 2198
375 M, %% M ↓ 2193
376 N, %% N ↙ 2199
377 O, %% O ← 2190
378 P, %% P ↖ 2196
379 Q, %% Q ↑ 2191
380 R, %% R ↗ 2197
381 S, %% S → 2192
382 T, %% T ↘ 2198
383 U, %% U ↓ 2193
384 V, %% V ↙ 2199
385 W, %% W ← 2190
```

```

386 X, %% X ↖ 2196
387 Y, %% Y ↑ 2191
388 Z, %% Z ↗ 2197
389 a, %% a right arrow top half 21C0
390 b, %% b left arrow top half 21BC
391 c, %% c → 2192
392 d, %% d ↘ 2198
393 e, %% e ↓ 2193
394 f, %% f ↙ 2199
395 g, %% g ← 2190
396 h, %% h ↖ 2196
397 i, %% i ↑ 2191
398 j, %% j ↗ 2197
399 k, %% k → 2192
400 l, %% l ↘ 2198
401 m, %% m ↓ 2193
402 n, %% n ↙ 2199
403 o, %% o ← 2190
404 p, %% p ↖ 2196
405 q, %% q ↑ 2191
406 r, %% r ↗ 2197
407 s, %% s → 2192
408 t, %% t ↘ 2198
409 u, %% u ↓ 2193
410 v, %% v ↙ 2199
411 w, %% w ← 2190
412 x, %% x ↖ 2196
413 y, %% y ↑ 2191
414 z, %% z ↗ 2197
415 }

```

`\l__adfarrows_tounicode_seq` A sequence to hold Unicode targets. These are not incredibly detailed, but hopefully more useful than PUA.

```

416 \seq_new:N \l__adfarrows_tounicode_seq
417 \seq_set_from_clist:Nn \l__adfarrows_tounicode_seq
418 {

```

outlines

```

419 21C0, %% A right arrow top half 21C0
420 21BC, %% B left arrow top half 21BC

```

outline shaft/tail with solid tip

```

421 2192, %% C → 2192
422 2198, %% D ↘ 2198
423 2193, %% E ↓ 2193
424 2199, %% F ↙ 2199
425 2190, %% G ← 2190
426 2196, %% H ↖ 2196
427 2191, %% I ↑ 2191
428 2197, %% J ↗ 2197

```

solid in various styles

```

429 2192, %% K → 2192
430 2198, %% L ↘ 2198
431 2193, %% M ↓ 2193
432 2199, %% N ↙ 2199
433 2190, %% O ← 2190
434 2196, %% P ↖ 2196
435 2191, %% Q ↑ 2191
436 2197, %% R ↗ 2197
437 2192, %% S → 2192
438 2198, %% T ↘ 2198
439 2193, %% U ↓ 2193
440 2199, %% V ↙ 2199
441 2190, %% W ← 2190
442 2196, %% X ↖ 2196
443 2191, %% Y ↑ 2191
444 2197, %% Z ↗ 2197
445 21C0, %% a right arrow top half 21C0
446 21BC, %% b left arrow top half 21BC
447 2192, %% c → 2192
448 2198, %% d ↘ 2198
449 2193, %% e ↓ 2193
450 2199, %% f ↙ 2199
451 2190, %% g ← 2190
452 2196, %% h ↖ 2196
453 2191, %% i ↑ 2191
454 2197, %% j ↗ 2197
455 2192, %% k → 2192
456 2198, %% l ↘ 2198
457 2193, %% m ↓ 2193
458 2199, %% n ↙ 2199
459 2190, %% o ← 2190
460 2196, %% p ↖ 2196
461 2191, %% q ↑ 2191
462 2197, %% r ↗ 2197
463 2192, %% s → 2192
464 2198, %% t ↘ 2198
465 2193, %% u ↓ 2193
466 2199, %% v ↙ 2199
467 2190, %% w ← 2190
468 2196, %% x ↖ 2196
469 2191, %% y ↑ 2191
470 2197, %% z ↗ 2197
471 }

472 \fixtounicode_tounicode:nnNN { ArrowsADF } { ArrowsADF }
473 \l__adfarrows_glyphtounicode_seq \l__adfarrows_tounicode_seq
474 }

```

__adfarrows_arrow:nn

```

475 \cs_new_nopar:Nn \__adfarrows_arrow:nn
476 {
477 \int_if_exist:cTF { l__adfarrows_base_#1_int }
478 {

```

```

479 \int_gset_eq:Nc \g__adfarrows_base_int { l__adfarrows_base_#1_int }
480 }{ % some kind of error check needed here
481 \int_gset:Nn \g__adfarrows_base_int { #1 }
482 }
483 \int_if_exist:cTF { l__adfarrows_dir_#2_int }
484 {
485 \int_gset_eq:Nc \g__adfarrows_add_int { l__adfarrows_dir_#2_int }
486 }{
487 \PackageError{adfarrows}{#2 not a valid direction. Setting east }
488 \int_gzero:N \g__adfarrows_add_int
489 }
490 \int_set:Nn \l__adfarrows_arrow_int { \g__adfarrows_base_int + \g__adfarrows_add_int
}
491 \int_compare:nNnTF { \l__adfarrows_arrow_int } < { 53 }
492 { % \int_compare:nNnTF { \l__adfarrows_arrow_int } > { 0 }
493 {
494 \expandafter\adfarrows@style\expandafter\char \int_to_arabic:n {
495 \l__adfarrows_arrow_int
496 }
497 }{
498 \PackageError{adfarrows}{\textbackslash l__adfarrows_arrow_int must
be greater than 0 but is \int_to_arabic:n {\l__adfarrows_arrow_int}}%
499 }
500 }{
501 \PackageError{adfarrows}{\textbackslash l__adfarrows_arrow_int must
be less than than 53 but is \int_to_arabic:n {\l__adfarrows_arrow_int}}%
502 }
503 }

```

__adfarrows_arrow:n

```

504 \cs_new_nopar:Nn \__adfarrows_arrow:n
505 {
506 \adfarrows@style\char#1
507 }

```

\adfarrows

```

508 \NewDocumentCommand \adfarrows { o m }
509 {
510 \group_begin:
511 \IfValueTF { #1 }
512 {
513 \__adfarrows_arrow:nn { #1 } { #2 }
514 }{
515 \__adfarrows_arrow:n { #2 }
516 }
517 \group_end:
518 }

519 \ExplSyntaxOff

```

\adfhalfarrowright

520 \newcommand*{\adhalfarrowright}{\adfarrow{1}}

\adhalfarrowleft

521 \newcommand*{\adhalfarrowleft}{\adfarrow{2}}

\adhalfarrowrightsolid

522 \newcommand*{\adhalfarrowrightsolid}{\adfarrow{27}}

\adhalfarrowleftsolid

523 \newcommand*{\adhalfarrowleftsolid}{\adfarrow{28}}

\adfarrowe

524 \gdef\adfarrowe#1{%
 525 \ifcase #1 \relax
 526 \or \adfarrow{3}%
 527 \or \adfarrow{11}%
 528 \or \adfarrow{19}%
 529 \or \adfarrow{29}%
 530 \or \adfarrow{37}%
 531 \or \adfarrow{45}%
 532 \fi}

\adfarrowse

533 \gdef\adfarrowse#1{%
 534 \ifcase #1 \relax
 535 \or \adfarrow{4}%
 536 \or \adfarrow{12}%
 537 \or \adfarrow{20}%
 538 \or \adfarrow{30}%
 539 \or \adfarrow{38}%
 540 \or \adfarrow{46}%
 541 \fi}

\adfarrows

542 \gdef\adfarrows#1{%
 543 \ifcase #1 \relax
 544 \or \adfarrow{5}%
 545 \or \adfarrow{13}%
 546 \or \adfarrow{21}%
 547 \or \adfarrow{31}%
 548 \or \adfarrow{39}%
 549 \or \adfarrow{47}%
 550 \fi}

\adfarrowsw

551 \gdef\adfarrowsw#1{%
 552 \ifcase #1 \relax

```

553 \or \adfarrow{6}%
554 \or \adfarrow{14}%
555 \or \adfarrow{22}%
556 \or \adfarrow{32}%
557 \or \adfarrow{40}%
558 \or \adfarrow{48}%
559 \fi}

```

\adfarrownw

```

560 \gdef\adfarrownw#1{%
561 \ifcase #1 \relax
562 \or \adfarrow{7}%
563 \or \adfarrow{15}%
564 \or \adfarrow{23}%
565 \or \adfarrow{33}%
566 \or \adfarrow{41}%
567 \or \adfarrow{49}%
568 \fi}

```

\adfarrown

```

569 \gdef\adfarrownw#1{%
570 \ifcase #1 \relax
571 \or \adfarrow{8}%
572 \or \adfarrow{16}%
573 \or \adfarrow{24}%
574 \or \adfarrow{34}%
575 \or \adfarrow{42}%
576 \or \adfarrow{50}%
577 \fi}

```

\adfarrown

```

578 \gdef\adfarrown#1{%
579 \ifcase #1 \relax
580 \or \adfarrow{9}%
581 \or \adfarrow{17}%
582 \or \adfarrow{25}%
583 \or \adfarrow{35}%
584 \or \adfarrow{43}%
585 \or \adfarrow{51}%
586 \fi}

```

\adfarrowne

```

587 \gdef\adfarrowne#1{%
588 \ifcase #1 \relax
589 \or \adfarrow{10}%
590 \or \adfarrow{18}%
591 \or \adfarrow{26}%
592 \or \adfarrow{36}%
593 \or \adfarrow{44}%
594 \or \adfarrow{52}%
595 \fi}

```

596 %% end adfarrows.sty

A.2 Font Definitions

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

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

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

```

598 \expandafter\ifx\csname adfarrows@scale\endcsname\relax
599 \let\adfarrows@scale\empty
600 \else
601 \edef\adfarrows@scale{s*[\csname adfarrows@scale\endcsname]}%
602 \fi
603 \DeclareFontFamily{U}{ArrowsADF}{}
604 \DeclareFontShape{U}{ArrowsADF}{m}{n}{
605 <-> \adfarrows@scale ArrowsADF
606 }{}
607 \DeclareFontShape{U}{ArrowsADF}{m}{sc}{<->ssub * ArrowsADF/m/n}{}
608 \DeclareFontShape{U}{ArrowsADF}{m}{it}{<->ssub * ArrowsADF/m/sc}{}
609 \DeclareFontShape{U}{ArrowsADF}{m}{sl}{<->ssub * ArrowsADF/m/it}{}
610 \DeclareFontShape{U}{ArrowsADF}{m}{si}{<->ssub * ArrowsADF/m/sl}{}
611 \DeclareFontShape{U}{ArrowsADF}{m}{scit}{<->ssub * ArrowsADF/m/si}{}
612 \DeclareFontShape{U}{ArrowsADF}{m}{scsl}{<->ssub * ArrowsADF/m/scit}{}
613 \DeclareFontShape{U}{ArrowsADF}{b}{n}{<->ssub * ArrowsADF/m/scsl}{}
614 \DeclareFontShape{U}{ArrowsADF}{b}{sc}{<->ssub * ArrowsADF/b/n}{}
615 \DeclareFontShape{U}{ArrowsADF}{b}{it}{<->ssub * ArrowsADF/b/sc}{}
616 \DeclareFontShape{U}{ArrowsADF}{b}{sl}{<->ssub * ArrowsADF/b/it}{}
617 \DeclareFontShape{U}{ArrowsADF}{b}{si}{<->ssub * ArrowsADF/b/sl}{}
618 \DeclareFontShape{U}{ArrowsADF}{b}{scit}{<->ssub * ArrowsADF/b/si}{}
619 \DeclareFontShape{U}{ArrowsADF}{b}{scsl}{<->ssub * ArrowsADF/b/scit}{}
620 \DeclareFontShape{U}{ArrowsADF}{bx}{n}{<->ssub * ArrowsADF/b/scsl}{}
621 \DeclareFontShape{U}{ArrowsADF}{bx}{sc}{<->ssub * ArrowsADF/bx/n}{}
622 \DeclareFontShape{U}{ArrowsADF}{bx}{it}{<->ssub * ArrowsADF/bx/sc}{}
623 \DeclareFontShape{U}{ArrowsADF}{bx}{sl}{<->ssub * ArrowsADF/bx/it}{}
624 \DeclareFontShape{U}{ArrowsADF}{bx}{si}{<->ssub * ArrowsADF/bx/sl}{}
625 \DeclareFontShape{U}{ArrowsADF}{bx}{scit}{<->ssub * ArrowsADF/bx/si}{}
626 \DeclareFontShape{U}{ArrowsADF}{bx}{scsl}{<->ssub * ArrowsADF/bx/scit}{}

627 \DeclareUnicodeCharacter{21C0}{right arrow top half}
628 \DeclareUnicodeCharacter{21BC}{left arrow top half}
629 \DeclareUnicodeCharacter{2192}{\textrightarrow}
630 \DeclareUnicodeCharacter{2198}{\searrow}
631 \DeclareUnicodeCharacter{2193}{\textdownarrow}
632 \DeclareUnicodeCharacter{2199}{\swarrow}
633 \DeclareUnicodeCharacter{2190}{\textleftarrow}
634 \DeclareUnicodeCharacter{2196}{\nrightarrow}
635 \DeclareUnicodeCharacter{2191}{\textuparrow}
636 \DeclareUnicodeCharacter{2197}{\nearrow}

```

adfsymbols: adfbullets

Clea F. Rees*

v1.5 (SVN Rev: 11700) 2026/02/25

```
637 \NeedsTeXFormat{LaTeX2e}
638 \RequirePackage{svn-prov}
639 \ProvidesPackageSVN[\filebase.sty]{$Id: adfbullets.dtx 11700 2026-02-25 07:11:48Z
  cfrees $}[v1.5 \revinfo]
640 \DefineFileInfoSVN[adfbullets]
641 \newif\if@adfbullets@digonnew
```

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

```
642 \@ifundefined{ExplLoaderFileDate}{%
643   \IfFileExists{expl3.sty}{%
644     \RequirePackage{expl3}%
645   }{%
646     \@adfbullets@digonnewfalse
647   }%
648 }{\@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.

```
649 \if@adfbullets@digonnew
650 \ExplSyntaxOn
651 \keys_define:nm { adfbullets }
652 {
653   scale .tl_set:N = \adfbullets@scale,
654   scale .initial:V = \@empty,
655 }
656 \else
657 \let\adfbullets@scale\@empty
658 \fi
```

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

```
659 %%%%%%%%%%%
660 \providecommand \IfFormatAtLeastTF { \@ifl@t@r \fmtversion }
```

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

```

661 \IfFormatAtLeastTF { 2022-06-01 }
662 {
663   \ProcessKeyOptions [ adfbullets ]
664 }{
665   \RequirePackage { l3keys2e }
666   \ProcessKeysOptions { adfbullets }
667 }
668 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
669 \ExplSyntaxOff

670 \RequirePackage{fixtounicode}

```

`\adfbullets@style`

```

671 \DeclareRobustCommand{\adfbullets@style}{%% do NOT break line below!
672   \not@math@alphabet\adfbullets@style\relax
673   \fontencoding{U}\fontfamily{BulletsADF}\fontseries{m}\fontshape{n}\selectfont
674 }

```

I don't know why somebody would use these fonts with a Unicode engine, but, just in case, map for that as well as pdfTeX.

```

675 \ExplSyntaxOn
676 \bool_if:nT { \sys_if_engine luatex_p: || \sys_if_engine pdftex_p: }
677 {

```

`_adfbullets_glyphunicode_seq` This seems ... insane?

It would be more efficient to just set everything directly, but this is easier to set up and only read once. First, a sequence to hold glyph names.

```

678 \seq_new:N \l__adfbullets_glyphunicode_seq
679 \seq_set_from_clist:Nn \l__adfbullets_glyphunicode_seq
680 {
681   A, %% A
682   B, %% B
683   C, %% C
684   D, %% D
685   E, %% E
686   F, %% F
687   G, %% G balloon 4-pointed asterisk 2724
688   H, %% H
689   I, %% I ☒ 2720 filled
690   J, %% J ☒ 2720 open
691   K, %% K
692   L, %% L
693   M, %% M
694   N, %% N
695   O, %% O
696   P, %% P
697   Q, %% Q
698   R, %% R
699   S, %% S
700   T, %% T

```

```

701 U, %% U
702 V, %% V
703 W, %% W
704 X, %% X
705 Y, %% Y 8-pointed rectilinear star 2737
706 Z, %% Z
707 a, %% a . filled 25CC
708 b, %% b . filled 25CC
709 c, %% c ■ 2B1B
710 d, %% d ◇ 2BC1
711 e, %% e ◁ 2BC7
712 f, %% f ▷ 2BC8
713 g, %% g triangle up 2BC5
714 h, %% h triangle down 2BC6
715 i, %% i arrowhead left top highlighted 2B98
716 j, %% j arrowhead right top highlighted 2B9A
717 k, %% k
718 l, %% l
719 m, %% m arrowhead left top highlighted 2B98 larger/darker
720 n, %% n arrowhead right top highlighted 2B9A larger/darker
721 o, %% o
722 p, %% p ellipse 2B2C
723 q, %% q dot large 25CE
724 r, %% r dot 00B7
725 s, %% s circled bullet 29BF circled bullet
726 t, %% t
727 u, %% u ■ 2BC0
728 v, %% v cusp 2BCC small
729 w, %% w cusp 2BCC med
730 x, %% x cusp 2BCC large
731 y, %% y cusp open 2BCE
732 z, %% z . open 25CB
733 }

```

`\l_adfbullets_tounicode_seq` A sequence to hold Unicode targets. These are not incredibly detailed, but hopefully more useful than none.

```

734 \seq_new:N \l_adfbullets_tounicode_seq
735 \seq_set_from_clist:Nn \l_adfbullets_tounicode_seq
736 {
737   2022 , %% A <fallback: text bullet>
738   2022 , %% B <fallback: text bullet>
739   2022 , %% C <fallback: text bullet>
740   2022 , %% D <fallback: text bullet>
741   2022 , %% E <fallback: text bullet>
742   2022 , %% F <fallback: text bullet>
743   2724 , %% G
744   2022 , %% H <fallback: text bullet>
745   2720 , %% I
746   2720 , %% J
747   2022 , %% K <fallback: text bullet>
748   2022 , %% L <fallback: text bullet>
749   2022 , %% M <fallback: text bullet>
750   2022 , %% N <fallback: text bullet>

```

```

751 2022 , %% O <fallback: text bullet>
752 2022 , %% P <fallback: text bullet>
753 2022 , %% Q <fallback: text bullet>
754 2022 , %% R <fallback: text bullet>
755 2022 , %% S <fallback: text bullet>
756 2022 , %% T <fallback: text bullet>
757 2022 , %% U <fallback: text bullet>
758 2022 , %% V <fallback: text bullet>
759 2022 , %% W <fallback: text bullet>
760 2022 , %% X <fallback: text bullet>
761 2737 , %% Y
762 2022 , %% Z <fallback: text bullet>
763 25CC , %% a
764 25CC , %% b
765 2B1B , %% c
766 2BC1 , %% d
767 2BC7 , %% e or 25C0 etc.?
768 2BC8 , %% f
769 2BC5 , %% g
770 2BC6 , %% h
771 2B98 , %% i
772 2B9A , %% j
773 2022 , %% k <fallback: text bullet>
774 2022 , %% l <fallback: text bullet>
775 2B98 , %% m
776 2B9A , %% n
777 2022 , %% o <fallback: text bullet>
778 2B2C , %% p
779 25CE , %% q
780 00B7 , %% r
781 29BF , %% s
782 25B0 , %% t
783 2BC0 , %% u
784 2BCC , %% v
785 2BCC , %% w
786 2BCC , %% x
787 2BCE , %% y
788 25CB , %% z
789 }

```

Generate the actual mappings.

```

790 \fixtounicode_tounicode:nnNN { BulletsADF } { BulletsADF }
791 \l__adfbullets_glyph_tounicode_seq \l__adfbullets_tounicode_seq
792 }
793 \ExplSyntaxOff

```

\adfbullet

```

794 \newcommand*\adfbullet[1]{\adfbullets@style\char#1}
795 %% end adfbullets.sty

```

A.3 Font Definitions

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

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

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

```

797 \expandafter\ifx\csname adfbullets@scale\endcsname\relax
798   \let\adfbullets@scale\@empty
799 \else
800   \edef\adfbullets@scale{s*[\csname adfbullets@scale\endcsname]}%
801   \fi
802 \DeclareFontFamily{U}{BulletsADF}{}
803 \DeclareFontShape{U}{BulletsADF}{m}{n}{
804   <-> \adfbullets@scale BulletsADF
805 }{}
806 \DeclareFontShape{U}{BulletsADF}{m}{sc}{<->ssub * BulletsADF/m/n}{}
807 \DeclareFontShape{U}{BulletsADF}{m}{it}{<->ssub * BulletsADF/m/sc}{}
808 \DeclareFontShape{U}{BulletsADF}{m}{sl}{<->ssub * BulletsADF/m/it}{}
809 \DeclareFontShape{U}{BulletsADF}{m}{si}{<->ssub * BulletsADF/m/sl}{}
810 \DeclareFontShape{U}{BulletsADF}{m}{scit}{<->ssub * BulletsADF/m/si}{}
811 \DeclareFontShape{U}{BulletsADF}{m}{scsl}{<->ssub * BulletsADF/m/scit}{}
812 \DeclareFontShape{U}{BulletsADF}{b}{n}{<->ssub * BulletsADF/m/scsl}{}
813 \DeclareFontShape{U}{BulletsADF}{b}{sc}{<->ssub * BulletsADF/b/n}{}
814 \DeclareFontShape{U}{BulletsADF}{b}{it}{<->ssub * BulletsADF/b/sc}{}
815 \DeclareFontShape{U}{BulletsADF}{b}{sl}{<->ssub * BulletsADF/b/it}{}
816 \DeclareFontShape{U}{BulletsADF}{b}{si}{<->ssub * BulletsADF/b/sl}{}
817 \DeclareFontShape{U}{BulletsADF}{b}{scit}{<->ssub * BulletsADF/b/si}{}
818 \DeclareFontShape{U}{BulletsADF}{b}{scsl}{<->ssub * BulletsADF/b/scit}{}
819 \DeclareFontShape{U}{BulletsADF}{bx}{n}{<->ssub * BulletsADF/b/scsl}{}
820 \DeclareFontShape{U}{BulletsADF}{bx}{sc}{<->ssub * BulletsADF/bx/n}{}
821 \DeclareFontShape{U}{BulletsADF}{bx}{it}{<->ssub * BulletsADF/bx/sc}{}
822 \DeclareFontShape{U}{BulletsADF}{bx}{sl}{<->ssub * BulletsADF/bx/it}{}
823 \DeclareFontShape{U}{BulletsADF}{bx}{si}{<->ssub * BulletsADF/bx/sl}{}
824 \DeclareFontShape{U}{BulletsADF}{bx}{scit}{<->ssub * BulletsADF/bx/si}{}
825 \DeclareFontShape{U}{BulletsADF}{bx}{scsl}{<->ssub * BulletsADF/bx/scit}{}

826 \DeclareUnicodeCharacter{2022}{\textbullet}
827 %%% \DeclareUnicodeCharacter{2724}{balloon 4-pointed asterisk}
828 % \DeclareUnicodeCharacter{2720}{\maltese$}
829 %%% \DeclareUnicodeCharacter{2737}{8-pointed rectilinear star}
830 % \DeclareUnicodeCharacter{25CC}{\circle{}} filled}
831 \DeclareUnicodeCharacter{2B1B}{\blacksquare$}
832 \DeclareUnicodeCharacter{2BC1}{\diamond$}
833 \DeclareUnicodeCharacter{2BC7}{\triangleleft$}
834 \DeclareUnicodeCharacter{2BC8}{\triangleright$}
835 \DeclareUnicodeCharacter{2BC5}{\triangle up}
836 \DeclareUnicodeCharacter{2BC6}{\triangle down}
837 \DeclareUnicodeCharacter{2B98}{\arrowhead left top highlighted}
838 \DeclareUnicodeCharacter{2B9A}{\arrowhead right top highlighted}
839 %% \DeclareUnicodeCharacter{2B2C}{\ellipse}
840 \DeclareUnicodeCharacter{25CE}{\dot large}
841 \DeclareUnicodeCharacter{00B7}{\dot}
842 \DeclareUnicodeCharacter{29BF}{\circled bullet}

```

```

843 % \DeclareUnicodeCharacter{2BC0}{$\blacksquare$}
844 \DeclareUnicodeCharacter{2BCC}{cusp}
845 \DeclareUnicodeCharacter{2BCE}{cusp open}
846 \DeclareUnicodeCharacter{25CB}{\circle{ } open}

```

Change History

v0.0	General: Correct prefix typo in docs.	19	Was <code>\adfarrows@fam{<}&{<}&{<}&{<}</code>	16
v1.2a	General: Fix lack of localisation bug.	1	<code>\adfarrows</code> : Remove pifont dependency.	19
			<code>\adfbullet</code> : Remove pifont dependency.	26
v1.2b	General: Include both PDF and TFM.	1	<code>\l__adfarrows_dir_west_int</code> : Try to make west arrows point west.	14
			<code>scale</code> :	12, 23
v1.3	General: May as well use <code>expl3</code> here. The alternative would be rewriting the code to use <code>TeX</code> counts, but for symbols like these there does not seem to be much reason to avoid the overhead of <code>expl3</code> . (Certainly almost anything would be an improvement over the current implementation, I suppose.)	13	<code>uarrwsadf.fd</code> : Support for scaling.	22
			<code>ubulletsadf.fd</code> : Support for scaling.	27
	Belated update for (N)NFSS (probably unneeded. Try switching to DTX/INS.	1	v1.4 General: Add <code>/ToUnicode</code> values (<code>adfarrows</code>).	16
			Add <code>/ToUnicode</code> values (<code>adfbullets</code>).	24
	Drop dependencies on <code>pifont</code> and <code>fp</code>	1	v1.5 General: Avoid broken implementation of <code>\pdfglyphtounicode</code> recommended in Lua <code>TeX</code> manual page 49.	16, 24
	Remove cack-handed dependency on <code>fp</code>	13	Use <code>fixtounicode</code>	13, 18, 24, 26
			v?? General: First public release.	1

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	<code>\@ifundefined</code>	264, 642
<code>\@adfarrows@digonnewfalse</code>	<code>__adfarrows_arrow:n</code>	504, 515
<code>\@adfarrows@digonnewtrue</code>	<code>__adfarrows_arrow:nn</code>	475, 513
<code>\@adfbullets@digonnewfalse</code>		
<code>\@adfbullets@digonnewtrue</code>		
<code>\@empty</code>	A	
<code>\@ifl@t@r</code>	<code>\adfarrows</code> 2, 3, 3, 508, 520, 521, 522, 523, 526, 527, 528, 529, 530,	

531, 535, 536, 537, 538, 539, 540, 544, 545, 546, 547, 548, 549, 553, 554, 555, 556, 557, 558, 562, 563, 564, 565, 566, 567, 571, 572, 573, 574, 575, 576, 580, 581, 582, 583, 584, 585, 589, 590, 591, 592, 593, 594	633, 634, 635, 636, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846
<code>\adfarrowe</code> 3, 524	<code>\def</code> 328
<code>\adfarrown</code> 3, 578	<code>\diamond</code> 832
<code>\adfarrowne</code> 3, 587	E
<code>\adfarrownw</code> 3, 569	<code>\edef</code> 601, 800
<code>\adfarrows</code> 3, 542	<code>\else</code> 278, 600, 656, 799
adfarrows (pkg.) 1	<code>\endcsname</code> 598, 601, 797, 800
<code>\adfarrows@scale</code> 599, 601, 605	<code>\expandafter</code> 494, 598, 797
<code>\adfarrows@scale</code> 275, 279	F
<code>\adfarrows@style</code> 292 , 494, 506	<code>\fi</code> 280, 532, 541, 550, 559, 568, 577, 586, 595, 602, 658, 801
<code>\adfarrows@west</code> 328	<code>\fixtounicode_tounicode:nnNN</code> 472, 790
<code>\adfarrowse</code> 3, 533	<code>\fmtversion</code> 282, 660
<code>\adfarrowsw</code> 3, 551	font definitions:
<code>\adfarroww</code> 3, 560	<code>uarrowsadf.fd</code> 597
<code>\adfbullet</code> 5, 794	<code>ubulletsadf.fd</code> 796
adfbullets (pkg.) 5	<code>\fontencoding</code> 294, 673
<code>\adfbullets@scale</code> 798, 800, 804	<code>\fontfamily</code> 294, 673
<code>\adfbullets@scale</code> 653, 657	<code>\fontseries</code> 294, 673
<code>\adfbullets@style</code> 671 , 794	<code>\fontshape</code> 294, 673
<code>\adfhalfarrowleft</code> 2, 521	G
<code>\adfhalfarrowleftsolid</code> 2, 523	<code>\g_adfarrows_add_int</code>
<code>\adfhalfarrowright</code> 2, 520 344 , 485, 488, 490
<code>\adfhalfarrowrightsolid</code> 2, 522	<code>\g_adfarrows_base_int</code>
B 343 , 479, 481, 490
<code>\blacksquare</code> 831, 843	<code>\gdef</code> 524, 533, 542, 551, 560, 569, 578, 587
<code>\bool_if:nT</code> 358, 676	<code>\group_begin:</code> 510
C	<code>\group_end:</code> 517
<code>\char</code> 494, 506, 794	I
<code>\circle</code> 830, 846	<code>\if@adfarrows@digonnew</code> 263, 271
<code>\cs_new_nopar:Nn</code> 475, 504	<code>\if@adfbullets@digonnew</code> ... 641, 649
<code>\csname</code> 598, 601, 797, 800	<code>\ifcase</code> 525, 534, 543, 552, 561, 570, 579, 588
D	<code>\IfFileExists</code> 265, 643
<code>\DeclareFontFamily</code> 603, 802	<code>\IfFormatAtLeastTF</code> 282, 283, 660, 661
<code>\DeclareFontShape</code> .. 604, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 803, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825	<code>\IfValueTF</code> 511
<code>\DeclareRobustCommand</code> 292, 671	<code>\ifx</code> 598, 797
<code>\DeclareUnicodeCharacter</code>	<code>\int_compare:nNnTF</code> 491, 492
627, 628, 629, 630, 631, 632,	<code>\int_gset:Nn</code> 481
	<code>\int_gset_eq:Nc</code> 479, 485
	<code>\int_gzero:N</code> 488
	<code>\int_if_exist:cTF</code> 477, 483
	<code>\int_new:N</code>
	298, 300, 302, 304, 306, 308, 310, 312, 314, 316, 318, 320, 322, 324, 326, 329, 331, 333,

335, 337, 339, 341, 343, 344, 345, 347, 349, 351, 353, 355, 357	
<code>\int_set:Nn</code>	299, 301, 303, 305, 307, 309, 311, 313, 315, 317, 319, 321, 323, 325, 327, 330, 332, 334, 336, 338, 340, 342, 346, 348, 350, 352, 354, 356, 490
<code>\int_to_arabic:n</code>	494, 498, 501
K	
<code>\keys_define:nn</code>	273, 651
L	
<code>\l__adfarrows_arrow_int</code>	357, 490, 491, 492, 495, 498, 501
<code>\l__adfarrows_base_c_int</code>	302
<code>\l__adfarrows_base_comic_int</code>	349
<code>\l__adfarrows_base_opentail_int</code>	345
<code>\l__adfarrows_base_ot_int</code>	298
<code>\l__adfarrows_base_p_int</code>	300
<code>\l__adfarrows_base_plain_int</code>	347
<code>\l__adfarrows_base_solidtail_int</code>	351
<code>\l__adfarrows_base_st_int</code>	304
<code>\l__adfarrows_base_t_int</code>	308
<code>\l__adfarrows_base_tail_int</code>	355
<code>\l__adfarrows_base_th_int</code>	306
<code>\l__adfarrows_base_thick_int</code>	353
<code>\l__adfarrows_dir_e_int</code>	310
<code>\l__adfarrows_dir_east_int</code>	312
<code>\l__adfarrows_dir_n_int</code>	335
<code>\l__adfarrows_dir_ne_int</code>	339
<code>\l__adfarrows_dir_north_int</code>	337
<code>\l__adfarrows_dir_northeast_int</code>	341
<code>\l__adfarrows_dir_northwest_int</code>	333
<code>\l__adfarrows_dir_nw_int</code>	331
<code>\l__adfarrows_dir_s_int</code>	318
<code>\l__adfarrows_dir_se_int</code>	314
<code>\l__adfarrows_dir_south_int</code>	320
<code>\l__adfarrows_dir_southeast_int</code>	316
<code>\l__adfarrows_dir_southwest_int</code>	324
<code>\l__adfarrows_dir_sw_int</code>	322
<code>\l__adfarrows_dir_w_int</code>	326
<code>\l__adfarrows_dir_west_int</code>	328
<code>\l__adfarrows_glyphunicode_seq</code>	360, 473
<code>\l__adfarrows_tounicode_seq</code>	416, 473
<code>\l__adfbullets_glyphunicode_seq</code>	678, 791
<code>\l__adfbullets_tounicode_seq</code>	734, 791
<code>\let</code>	279, 599, 657, 798
M	
<code>\maltese</code>	828
N	
<code>\nearrow</code>	636
<code>\newcommand</code>	520, 521, 522, 523, 794
<code>\NewDocumentCommand</code>	508
<code>\newif</code>	263, 641
<code>\not@math@alphabet</code>	293, 672
<code>\nwarrow</code>	634
O	
options:	
scale	2, 5, 271, 649
<code>\or</code>	526, 527, 528, 529, 530, 531, 535, 536, 537, 538, 539, 540, 544, 545, 546, 547, 548, 549, 553, 554, 555, 556, 557, 558, 562, 563, 564, 565, 566, 567, 571, 572, 573, 574, 575, 576, 580, 581, 582, 583, 584, 585, 589, 590, 591, 592, 593, 594
P	
<code>\PackageError</code>	487, 498, 501
packages:	
adfarrows	1
adfbullets	5
<code>\ProcessKeysOptions</code>	288, 666
<code>\providecommand</code>	282, 660
<code>\ProvidesFile</code>	597, 796
R	
<code>\relax</code>	293, 525, 534, 543, 552, 561, 570, 579, 588, 598, 672, 797
S	
scale (opt.)	2, 5, 271, 649
<code>\searrow</code>	630
<code>\selectfont</code>	294, 673
<code>\seq_new:N</code>	360, 416, 678, 734
<code>\seq_set_from_clist:Nn</code>	361, 417, 679, 735
<code>\swarrow</code>	632
<code>\sys_if_engine_luatex_p:</code>	358, 676
<code>\sys_if_engine_pdftex_p:</code>	358, 676
T	
<code>\textbackslash</code>	498, 501
<code>\textbullet</code>	826
<code>\textdownarrow</code>	631
<code>\textleftarrow</code>	633
<code>\textrightarrow</code>	629
<code>\textuparrow</code>	635
<code>\triangleleft</code>	833
<code>\triangleright</code>	834

	U			
uarrrowsadf.fd (fd.)	<u>597</u>		
		ubulletsadf.fd (fd.)	<u>796</u>