

The `erw-l3` package ^{*}

Erwann Rogard[†]

Released 2020/05/23

Abstract

Utilities like `expl3[1]`.

Résumé

Utilitaires de type `expl3[1]`.

Contents

I	Usage	4
1	Loading the package	4
2	<code>cs</code>	4
3	<code>csint</code>	4
4	<code>int</code>	5
5	<code>keys</code>	5
6	<code>lambda</code>	5
7	<code>option</code>	5
8	<code>prop</code>	5
9	<code>seq</code>	6
10	<code>sys</code>	6
11	<code>tl</code>	6
II	Listing	8

^{*}This file describes version v2.8, last revised 2020/05/23.

[†]firstname dot lastname AusTria gmail dot com

1	constants	8
	1.	8
2	csint	8
	2.	8
3	int	8
	3.	8
4	lambda	9
	4.	9
5	prop	9
	5.	9
	6.	9
	7.	9
6	seq	10
	8.	10
	9.	10
	10.	10
	11.	11
7	sys	11
	12.	11
	13.	12
8	tl	12
	14.	12
	15.	12
	17.	13
	18.	13
	19.	13
	20.	14
	21.	14
	22.	15
III	Other	16
1	Acknowledgment	16
2	Install	16
3	Support	16
	3.1 Platform	16
	3.2 Engine	16
	3.3 Results	16
4	References	16

Change History	17
Index	18
IV Implementation	22
1 Opening	22
2 cs	22
2.1 backend	22
2.2 frontend	22
3 csint	22
3.1 backend	22
3.2 frontend	23
4 int	24
4.1 backend	24
4.2 frontend	24
5 keys	25
5.1 frontend	25
6 lambda	25
7 msg	25
7.1 backend	25
7.2 frontend	25
8 prop	26
8.1 backend	26
8.2 frontend	26
9 oper	27
9.1 backend	27
9.2 frontend	27
10 option	27
11 seq	27
11.1 backend	27
11.2 frontend	28
12 sys	28
12.1 backend	28
13 tl	31
13.1 backend	31
13.2 frontend	32

Part I

Usage

<code>\usepackage</code>	<code>\usepackage{erw-l3}</code>
--------------------------	----------------------------------

Requirement

1. `erw-l3.sty` and its dependencies are in the path of the \LaTeX engine. See [Part III, section 3](#).
2. Goes in the *preamble*

2 cs

<code>\erw_cs_identity:n</code>	<code>\erw_cs_identity:n{<arg>}</code>
---------------------------------	----------------------------------------------

<code>\erw_cs_set_inline:Nn</code>	<code>\erw_cs_set_inline:Nn<cs>{<code>}</code>
<code>\erw_cs_set_inline:(cn cn)</code>	
<code>\erw_cs_gset_inline:Nn</code>	

3 csint

<code>\erw_csint:nn</code>	<code>\erw_csint:nn{<integer>}{<arg>}</code>
----------------------------	----------------------------------------------------------

<code>\erw_csint_name:n</code>	<code>\erw_csint_name:n{<integer>}</code>
--------------------------------	-------------------------------------------------

<code>\erw_csint_names:nnn</code>	<code>\erw_csint_names:nnn{<integer>}{<integer>}{<integer>}</code>
-----------------------------------	--------------------------------------------------------------------------------------

<code>\erw_csint_names_braced:</code>	
<code>\erw_csint_names_braced:n</code>	
<code>\erw_csint_names_braced:nnn</code>	

<code>\erw_csint_new:n</code>	<code>\erw_csint_new:n{<integer>}</code>
-------------------------------	------------------------------------------------

<code>\erw_csint_reset:</code>	<code>\erw_csint_reset:</code>
--------------------------------	--------------------------------

4 int

<code>\erw_int_range:n</code>	<code>\erw_int_range:n{<integer>}</code>
<code>\erw_int_range:nn</code>	

5 keys

<code>\erw_keyval_error:Nn</code>	<code>\erw_keyval_error:Nn<token>{<keyval list>}</code>
<code>\erw_keyval_error:Nnn</code>	<code>\erw_keyval_error:Nnn<token>{<clist>}</code>

6 lambda

<code>\erw_lambda:nnn</code>	<code>\erw_lambda:nnn<token>{<arg spec>}{<code>}</code>
------------------------------	---------------------------------------------------------------------------

7 option

<code>\erw_option:n</code>	<code>\erw_option:n{<keyval list>}</code>
----------------------------	-------------------------------------------------

oper / fold_set_par
oper / fold_apply_par
sys / timestamp_delim

8 prop

All functions that modify a $\langle prop \rangle$ check it exists, if not make sure it does.

<code>\erw_prop_keyval_parse:NNNn</code>	<code>\erw_prop_keyval_parse:NNNn<prop><cs1><cs2>{<keyval list>}</code>
------------------------------------------	-------------------------------------------------------------------------------------------------

<code>\erw_prop_map_item:NNN</code>	<code>\erw_prop_map_item:NNN<cs><prop1><prop2></code>
-------------------------------------	-------------------------------------------------------------------------

<code>\erw_prop_to_clist:Nn</code>	<code>\erw_prop_to_clist:Nn<prop>{<key1>,...}</code>
------------------------------------	------------------------------------------------------------------

9 seq

All functions that modify a $\langle seq \rangle$ check it exists, if not make sure it does.

<hr/> <hr/>	$\backslash erw_seq_compose:nN$	$\backslash erw_seq_compose:nN\{\{\langle cs_1 \rangle\} \dots\} \langle seq \rangle$
<hr/> <hr/>	$\backslash erw_seq_compose_c:nN$	$\backslash erw_seq_compose_c:nN\{\{\langle cs\ name_1 \rangle\} \dots\} \langle seq \rangle$
<hr/> <hr/>	$\backslash erw_seq_compose_vers:nN$	$\backslash erw_seq_compose:nN\{\{\langle cs\ or\ code_1 \rangle\} \dots\} \langle seq \rangle$
<hr/> <hr/>	$\backslash erw_seq_put_right_clist:Nn$ $\backslash erw_seq_put_right_clist:cn$	$\backslash erw_seq_put_right_clist:Nn \langle seq \rangle \{\langle clist \rangle\}$
<hr/> <hr/>	$\backslash erw_seq_put_right_prop:Nnn$	$\backslash erw_seq_put_right_prop:Nnn \langle seq \rangle \langle prop \rangle \{\langle clist \rangle\}$
<hr/> <hr/>	$\backslash erw_seq_use:Nn$	$\backslash erw_seq_use:Nn \langle seq \rangle \{\langle items \rangle\}$
	Also see [1, Section 8 of l3seq]	
	Semantics $\backslash seq_use:Nnnn \langle seq \rangle \backslash erw_tl_separators:n \{\langle items \rangle\}$	

10 sys

<hr/> <hr/>	$\backslash erw_sys_jobnametimestamp:nn$ $\backslash erw_sys_jobnametimestamp:$	$\backslash erw_sys_jobnametimestamp:nn \{date time datetime\} \{10 16\}$
<hr/> <hr/>	$\backslash erw_sys_timestamp:nn$ $\backslash erw_sys_timestamp:$	$\backslash erw_sys_timestamp:nn \{date time datetime\} \{10 16\}$ Semantics Timestamp in base 10 or 16
<hr/> <hr/>	$\backslash erw_sys_timestamp_delimiter:$	$\backslash erw_sys_timestamp_delimiter:$

11 tl

All functions that modify a $\langle token\ list \rangle$ check it exists, if not make sure it does.

<hr/> <hr/>	$\backslash erw_tl_append_item:nn$	$\backslash erw_tl_append_item:nn \{\langle arg\ list \rangle\} \{\langle arg \rangle\}$
<hr/> <hr/>	$\backslash erw_tl_compose:nN$ $\backslash erw_tl_compose:nn$	$\backslash erw_tl_compose:nn \{\{cs_1\} \dots\} \{\langle token\ list \rangle\}$

<u>\erw_tl_compose_c:nN</u> <u>\erw_tl_compose_c:nn</u>	\erw_tl_compose_c:nn{cs name_1}...{token list}
<u>\erw_tl_compose_vers:nN</u> <u>\erw_tl_compose_vers:nn</u>	\erw_tl_compose_vers:nn{cs or code_1}...{token list}
<u>\erw_tl_fold:NN</u> <u>\erw_tl_fold:cN</u>	\erw_tl_fold:NN⟨cs⟩⟨tl var⟩
<u>\erw_tl_gset_function:N</u> <u>\erw_tl_gset_function:n</u>	\erw_tl_gset_function:n{code}
<u>\erw_tl_join:nn</u> <u>\erw_tl_join:nnn</u> <u>\erw_tl_join:nnnn</u> <u>\erw_tl_join:nnnnn</u>	\erw_tl_join:nn{token list ₁ }{token list ₂ }
<u>\erw_tl_last_item:n</u>	\erw_tl_last_item:n{token list}
<u>\erw_tl_map:n</u> <u>\erw_tl_map:Nn</u>	\erw_tl_map:n{items}
	Semantics Maps over ⟨items⟩ using the internal function set by <code>\erw_tl_gset_function:n</code>
<u>\erw_tl_map_inline:nn</u>	\erw_tl_map_inline:nn{code}{items}
<u>\erw_tl_map_thread:Nn</u>	\erw_tl_math_thread:Nn⟨cs⟩{items}
<u>\erw_tl_map_thread_at:Nnn</u>	\erw_tl_math_thread_at:Nnn{integer}{token list}
<u>\erw_tl_repeat:nn</u>	\erw_tl_repeat:nn{integer}{token list}
<u>\erw_tl_split:nnn</u> <u>\erw_tl_split:nn</u>	\erw_tl_split:nn{items}{delimiter}
<u>\erw_tl_separators:n</u>	\erw_tl_separators:n{items}
	Semantics According to the count of ⟨items⟩: <ol style="list-style-type: none"> 1) {token list₁}{token list₁}{token list₁} 2) {token list₁}{token list₂}{token list₁ token list₂} 3) {token list₁}{token list₂}{token list₃}

Part II

Listing

1 constants

Listing 1.

```
\ExplSyntaxOn
\seq_const_from_clist:Nn \foo_seq{ A, B, C }
\prop_const_from_keyval:Nn \foo_prop{ A = a, B = b, C = c }
\ExplSyntaxOff
```

2 csint

Listing 2.

```
\ExplSyntaxOn
\cs_set:Nn\__foo:n{ f(#1) }
\cs_set:Nn\__baz:n{ h\{#1\} }
\tl_map_function:nN { {\__baz:n} {g[#1]} {\__foo:n} }\erw_csint_new:n
\exp_last_unbraced:Nx
\erw_tl_compose_c:nn
{\erw_csint_names_braced:nnn{ 1 }{ 1 }{ 3 }}
{ X }}
\ExplSyntaxOff
```

$h\{g[f(X)]\}$

3 int

Listing 3.

```
\ExplSyntaxOn
\erw_int_range:nn{ 2 }{ 5 }\\
\erw_int_range:n{ 5 }
\ExplSyntaxOff
```

2345
12345

4 lambda

Listing 4.

```
\ExplSyntaxOn
\tl_set:Nn \l_tmpa_tl
{
  \erw_lambda:nnn \DeclareDocumentCommand{ m }{ Hello,~#1! }
}
\l_tmpa_tl{ world }
\ExplSyntaxOff
```

Hello, world!

5 prop

Listing 5.

```
\ExplSyntaxOn
\erw_prop_map_item:NNN \prop_put:Nnx \baz_prop \foo_prop
\prop_if_exist:NTF\baz_prop{T}{F}\
\prop_item:Nn \baz_prop{ A }
,\prop_item:Nn \baz_prop{ B }
,\prop_item:Nn \baz_prop{ C }
\ExplSyntaxOff
```

T
a,b,c

Listing 6.

```
\ExplSyntaxOn
\erw_prop_keyval_parse:NNNn
\foo_prop
\erw_keyval_error:Nn
\prop_put:Nnn{ X = x, Y = y, Z = z }
\prop_item:Nn \foo_prop{ X }
,\prop_item:Nn \foo_prop{ Y }
,\prop_item:Nn \foo_prop{ Z }
\ExplSyntaxOff
```

x,y,z

Listing 7.

```
\ExplSyntaxOn
\erw_prop_to_clist:Nn \foo_prop{ A, B, C }
\ExplSyntaxOff
```

a,b,c

6 seq

Listing 8.

```
\ExplSyntaxOn
\cs_set:Nn \__foo:n { f(#1) }
\cs_set:Nn \__bar:n { g[#1] }
\cs_set:Nn \__baz:n { h\{#1\} }
\seq_new:N \l_tmp_seq
\seq_put_right:Nn \l_tmp_seq{X}
\erw_seq_compose:nN{ \__baz:n}{ \__bar:n}{ \__foo:n} \l_tmp_seq
\seq_item:Nn \l_tmp_seq{ 1 }\\
\seq_item:Nn \l_tmp_seq{ 2 }\\
\seq_item:Nn \l_tmp_seq{ 3 }\\
\seq_item:Nn \l_tmp_seq{ 4 }
\ExplSyntaxOff
```

X
f(X)
g[f(X)]
h{g[f(X)]}

Listing 9.

```
\ExplSyntaxOn
\cs_set:Nn \__foo:n { f(#1) }
\cs_set:Nn \__bar:n { g[#1] }
\cs_set:Nn \__baz:n { h\{#1\} }
\seq_put_right:Nn \l_tmpa_seq{X}
\erw_seq_compose_c:nN{ \__baz:n}{ \__bar:n}{ \__foo:n} \l_tmpa_seq
\seq_item:Nn \l_tmpa_seq{ 1 }\\
\seq_item:Nn \l_tmpa_seq{ 2 }\\
\seq_item:Nn \l_tmpa_seq{ 3 }\\
\seq_item:Nn \l_tmpa_seq{ 4 }
\ExplSyntaxOff
```

X
f(X)
g[f(X)]
h{g[f(X)]}

Listing 10.

```
\ExplSyntaxOn
\erw_seq_put_right_prop:Nn \bar_seq\foo_prop{ A, B, C }
\seq_use:Nn\bar_seq{,}
\ExplSyntaxOff
```

a,b,c

Listing 11.

```
\ExplSyntaxOn
\seq_put_right:Nn\l_tmpa_seq{ A }
\seq_put_right:Nn\l_tmpa_seq{ B }
\erw_seq_use:Nn \l_tmpa_seq{ {~and~} }\\
\erw_seq_use:Nn \l_tmpa_seq{ {,\ }{~and~} }\\
\erw_seq_use:Nn \l_tmpa_seq{ {~and~}{,\ }{~and~} }\\[1em]
\seq_put_right:Nn\l_tmpa_seq{ C }
\erw_seq_use:Nn \l_tmpa_seq{ {~and~} }\\
\erw_seq_use:Nn \l_tmpa_seq{ {,\ }{and~} }\\
\erw_seq_use:Nn \l_tmpa_seq{ {~and~}{,\ }{~and~} }\\
\ExplSyntaxOff
```

A and B

A and B

A and B

A and B and C

A, B, and C

A, B, and C

7 sys

Listing 12.

```
\ExplSyntaxOn
\noindent\erw_sys_timestamp:nn{date}{10}{-}
\noindent\erw_sys_timestamp:nn{time}{10}\\
\noindent\erw_sys_timestamp:nn{datetime}{10}\\
\erw_sys_timestamp:nn{date}{16}{\%}
\erw_sys_timestamp:nn{time}{16}\\
\erw_option:n{ sys / timestamp_delim = {\%} }
\erw_sys_timestamp:nn{datetime}{16}\\
\erw_sys_jobnametimestamp:
\ExplSyntaxOff
```

```

20200525-258
20200525-258
1343c4d%102
1343c4d%102
erw-l3%1343c4d%102

```

Listing 13.

```

\ExplSyntaxOn
\erw_option:n{ sys / timestamp_delim = \c_empty_tl }
\iow_new:N \foo_iow
\tl_set:Nx \foo_dec { \erw_sys_timestamp:nn{datetime}{10} }
\tl_set:Nx \foo_hex { \erw_sys_timestamp: }
\iow_open:Nn \foo_iow{ \foo_hex }
\iow_now:Nn\foo_iow{ Hello,\ world! }
\iow_close:N \foo_iow
D:\foo_dec\\
\file_timestamp:n{ \foo_hex }\\
\file_input:n{ \foo_hex }
\ExplSyntaxOff

```

```

D:20200525258
D:20200525025839-04'00'
Hello, world!

```

8 tl

Listing 14.

```

\ExplSyntaxOn
\cs_set:Nn \__foo:n { f(#1) }
\cs_set:Nn \__bar:n { g[#1] }
\cs_set:Nn \__baz:n { h\{#1\} }
\tl_set:Nn \l_tmpa_tl{ X }
\erw_tl_compose:nN{ {\__baz:n}{\__bar:n}{\__foo:n} }\l_tmpa_tl
\l_tmpa_tl\\
\tl_set:Nn \l_tmpa_tl{ X }
\erw_tl_compose:nn{ {\__baz:n}{\__bar:n}{\__foo:n}}{ X }\\
\ExplSyntaxOff

```

```

h{g[f(X)]}
h{g[f(X)]}

```

Listing 15.

```
\ExplSyntaxOn
\cs_set:Nn \__foo:n { f(#1) }
\cs_set:Nn \__bar:n { g[#1] }
\cs_set:Nn \__baz:n { h\{#1\} }
\tl_set:Nn \l_tmpa_tl{ X }
\erw_tl_compose_c:nN{ \__baz:n\__bar:n\__foo:n } \l_tmpa_tl
\l_tmpa_tl\
\erw_tl_compose_c:nn{ \__baz:n\__bar:n\__foo:n}{X }
\ExplSyntaxOff
```

```
h{g[f(X)]}
h{g[f(X)]}
```

Listing 16.

```
\ExplSyntaxOn
\cs_set:Npn \__foo #1 { f(#1) }
\cs_set:Npn \__bar #1 { g[#1] }
\cs_set:Npn \__baz #1 { h\{#1\} }
\erw_tl_compose_vers:nn{ {\__baz}{g[#1]}\__foo}{X }
\ExplSyntaxOff
```

```
h{g[f(X)]}
```

Listing 17.

```
\ExplSyntaxOn
\cs_set:Nn \__foo:n { f(#1) }
\tl_set:Nn \l_tmpa_tl{ X }
\erw_tl_fold:NN\__foo:n\l_tmpa_tl
\l_tmpa_tl\
\cs_set:Nn \__bar:n { g[#1] }
\erw_tl_fold:cN {\__bar:n}\l_tmpa_tl
\l_tmpa_tl
\ExplSyntaxOff
```

```
f(X)
g[f(X)]
```

Listing 18.

```
\ExplSyntaxOn
\erw_tl_repeat:nn{ 3 }{ x }
\ExplSyntaxOff
```

```
xxx
```

Listing 19.

```
\ExplSyntaxOn
\erw_tl_split:nn{ {a} {b} {c} }{ == }
\ExplSyntaxOff
```

a==b==c

Listing 20.

```
\ExplSyntaxOn
\cs_set:Nn \__foo:n { (#1) }
\erw_tl_map:Nn \__foo:n{ {a}{b}{c} }
\ExplSyntaxOff
```

(a)(b)(c)

Listing 21.

```
\ExplSyntaxOn
\cs_set:Nn \__foo:n { (#1) }
\erw_tl_map_thread:Nn \__foo:n
{
  { {a}{b}{c}{d}{e}{f} }
}\
\cs_set:Nn \__foo:nn { (#1+#2) }
\erw_tl_map_thread:Nn \__foo:nn
{
  { {a}{b}{c}{d}{e}{f} }
  { {A}{B}{C}{D}{E}{F} }
}\
\cs_set:Nn \__foo:nnn { (#1+#2+#3) }
\erw_tl_map_thread:Nn \__foo:nnn
{
  { {a}{b}{c}{d}{e}{f} }
  { {A}{B}{C}{D}{E}{F} }
  { {k}{l}{m}{n}{o}{p} }
}\
\cs_set:Nn \__foo:nnnn { (#1+#2+#3+#4) }
\erw_tl_map_thread:Nn \__foo:nnnn
{
  { {a}{b}{c}{d}{e}{f} }
  { {A}{B}{C}{D}{E}{F} }
  { {k}{l}{m}{n}{o}{p} }
  { {K}{L}{M}{N}{O}{P} }
}
\ExplSyntaxOff
```

(a)(b)(c)(d)(e)(f)
(a+A)(b+B)(c+C)(d+D)(e+E)(f+F)

(a+A+k)(b+B+l)(c+C+m)(d+D+n)(e+E+o)(f+F+p)
(a+A+k+K)(b+B+l+L)(c+C+m+M)(d+D+n+N)(e+E+o+O)(f+F+p+P)

Listing 22.

```
\ExplSyntaxOn
\cs_set:Nn\__foo:nn { (#1+#2) }
\erw_tl_map_thread_at:Nnn \__foo:nn{ 2 }
{
  { {a}{b}{c}{d}{e}{f} }
  { {A}{B}{C}{D}{E}{F} }
}
\ExplSyntaxOff
```

(b+B)

Part III

Other

1 Acknowledgment

This work has benefited from Q&A's from the L^AT_EX community [3]. `lambda` originally appeared in [2].

2 Install

- 1) Compile `erw-13.dtx` (under Unix, `$tex timestamp.dtx`)
- 2) Put the generated `erw-13.sty` in the search path of the L^AT_EX engine

3 Support

This package is available from <https://www.ctan.org/pkg/erw-13> and <https://github.com/rogard/erw-13>.

3.1 Platform

- i) Linux laptop 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24
↪ 06:16:15 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux

3.2 Engine

- a) pdfTeX 3.14159265-2.6-1.40.20 (TeX Live 2019)
- b) pdfTeX 3.14159265-2.6-1.40.21 (TeX Live 2020)
- c) LuaHBTeX, Version 1.12.0 (TeX Live 2020)
- d) XeTeX 3.14159265-2.6-0.999992 (TeX Live 2020)

3.3 Results

- 1) `erw-13 v2.0` compiles satisfactorily on platform *i)* and engines *b)*, *c)*, and *d)*

References

- [1] The L^AT_EX3 Project Team *The L^AT_EX3 interfaces*, 2019, <http://ftp.math.purdue.edu/mirrors/ctan.org/macros/latex/contrib/l3kernel/interface3.pdf>
- [2] @sean-allred's answer to "How to create lambda expressions?", <https://tex.stackexchange.com/a/188053/112708>
- [3] <https://tex.stackexchange.com/users/112708/erwann?tab=questions>

Change History

v1.1	General: \numbrdcsnew changed to \newnumbrdcs and made 'disambiguable' 15	v1.6	General: Fix: critical bug preventing erw-l3 from working without explicit inclusion of expl3 15
	disambig/backend: changes to the key, added \ProcessPackageKeysOption; . . . 15	v1.7	General: Add: option 15
	Brought all the modules under one file; renamed l3erw to erw-l3; . . . 15		Add: sys 15
v1.2	General: disambig: \disambignewcmd no longer takes a token name as arg, rather a token. 15		Move: \erw_fold_apply_par:n . . 15
	disambig: pushed the code inside \keys_define; 15		Move: \erw_fold_set_par:n . . . 15
	Add: \erw_items_to 15		Rearrange: structure of implementation, e.g. section 9 . . . 15
	Add: \erw_last_item 15		Remove: document level functions, \numbrdcsnew, \numbrdcs 15
	Add: \erw_repeat 15		Replace: listing's implem with that of tocloft 15
	Add: \erw_split 15		Replace: vers. numb. from 3 to 2 digits 15
	Add: \map_thread 15	v1.8	General: (deleted) 15
	Front end cmds no longer generated with module disambig; Option of the same name deleted; 15		Add: function for all frontend functions. 15
	Modify: \erw_compose, order in which functions composed ($g \circ f$ means f comes before g) 15		Remove: \erw_cs_set_eq:NN and variants 15
	Rearrange: the doc to clearly separate frontend from backend . . 15		Remove: \erw_is_matrix:n (predicate must be expandable) . . 15
v1.3	General: Replace: versioning, should have been 0.1.2 15		Rename: all cs prefixes to agree with heading under which they come, e.g. \erw_identity:n by \erw_cs_identity:n 15
v1.4	General: Add: \erw_accum 15		Replace: \erw_seq_fold:NN by \erw_oper_fold_seq:NN and likewise for variants 15
	Add: \erw_int_range 15	v1.9	General: Add: \erw_sys_timestamp_delimiter: 15
	Add: \erw_is_matrix (to check arg of \erw_tl_map_thread:Nn) 15		Add: \erw_tl_join:nn and variants 15
	Add: \erw_merge 15		Rename: \erw_append_arg:nn to \erw_tl_append_item:nn 15
	Add: \erw_set_map_inline 15		Rename: \erw_oper_gset_function:N to \erw_tl_gset_function:N (and variants) 15
	Add: \erw_set_map 15	v2.0	General: Add: \erw_jobnametimestamp:nn and variants 15
	Remove: \erw_items_to (redundant with \tl_range:nnn) . 15		Remove: \merge:nn (redundant with \erw_join:nn) 15
v1.5	General: Modify: source repository . . 15		Rename: v0.0 to v1.0, etc. 15
	Rearrange: frontend/backend sections 15		
	Remove: disambig 15		
	Split Section Preliminaries into Conventions and Requirement. . . 15		

v2.1	General: Add: \erw_prop_to_clist:Nn, \erw_prop_put:NN, and \erw_prop_put:Nnn 15 Add: \erw_seq_from_clist:Nn, \erw_seq_from_prop:NNn, and \erw_seq_put_right:Nn 15 Move: all functions under section 9 to section 13 or section 11 , except \@@oper_compose:NnN 15 Replace: \erw_seq_fold:NN by __erw_seq_fold:NN 15	v2.6	General: Add: \erw_cs_error:nn ... 15 Add: \erw_cs_error:n 15 Add: \erw_keyval_parse:NNNn .. 15 Add: \erw_prop_keyval_parse:NNNn .. 15 Add: \erw_prop_map_item:NNN .. 15 Add: \msg_new:nnn{erw}{varnset} 15 Remove: \erw_cs_apply 15 Remove: \erw_prop_put:NN 15 Remove: \erw_prop_put_keyval:Nn 15 Remove: \msg_new:nnn, module erw, messages: keyval/... 15 Rename: basics to cs 15 Replace: \erw_seq_from_clist by \erw_seq_put_right_clist 15 Replace: \erw_seq_from_prop by \erw_seq_put_right_prop 15
v2.2	General: Add: \erw_seq_use:Nn 15 Add: \erw_tl_separators:n 15	v2.7	General: Add: \erw_keyval_error:Nnn 15 Add: \erw_keyval_error:Nn 15 Remove: \erw_cs_error:nn 15 Remove: \erw_cs_error:n 15
v2.3	General: Add: \msg_new:nnn{erw}{csnset} 15 Add: \msg_new:nnn{erw}{keyval/...} . 15 Fix: 'mark as private code' (hitherto unnoticed) 15 Modify: behavior of \erw_seq_use:Nn 15 Move: all \msg_new:Nnnn statements under same heading .. 15	v2.8	General: Add: \msg_new:nnn{erw}{notset} 15 Remove: \msg_new:nnn{erw}{csnset} 15 Remove: \msg_new:nnn{erw}{varnset} ... 15
v2.4	General: Add: \erw_lambda:nnn 15		
v2.5	General: Add: \erw_prop_put_keyval:Nn 15		

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

B	415, 421, 425, 434, 443, 447, 451, 459, 460, 461, 462, 477, 481, 492, 527
\begin	320
C	
cs commands:	
\cs_generate_variant:Nn ..	13, 18, 133, 190, 196, 215, 222, 229, 396, 442
\cs_gset:Npn	16
\cs_new:Nn	4, 9, 14, 21, 25, 26, 41, 46, 47, 77, 81, 87, 88, 152, 197, 201, 205, 223, 230, 236, 245, 246, 247, 255, 256, 266, 267, 278, 279, 280, 286, 292, 298, 321, 322, 323, 327, 331, 374, 397, 401, 405, 411,
	415, 421, 425, 434, 443, 447, 451, 459, 460, 461, 462, 477, 481, 492, 527
\cs_new_protected:Nn	30, 51, 89, 113, 121, 134, 143, 174, 179, 191, 209, 216, 303, 336, 463, 467, 472, 496, 517
\cs_new_protected:Npn	98
\cs_set:Nn	123
\cs_set:Npn	8, 11, 56
\cs_set_eq:NN	469
\cs_set_protected:Nn	91, 92, 115, 181, 343, 348, 353, 359, 366
\cs_split_function:N	6

E

erw commands:

`\erw_cs_gset_eq:Nn` 445
`\erw_cs_gset_inline:Nn` .. 2, 9, 14, 449
`\erw_cs_identity:n` 2, 8
`\erw_cs_set_inline:Nn` 2, 9, 33, 154, 474
`\erw_csint:nn` 3, 21
`\erw_csint_name:n` ... 3, 20, 25, 28, 46
`\erw_csint_names:nnn` 3, 26
`\erw_csint_names_braced:` .. 3, 41, 431
`\erw_csint_names_braced:n` 3, 41
`\erw_csint_names_braced:nnn` .. 3, 41
`\erw_csint_new:n` 3, 30, 428
`\erw_csint_reset:` 3, 51, 427
`\erw_int_range:n` 3, 77
`\erw_int_range:nn` 3, 77
`\erw_keyval_error:Nn` 3, 87
`\erw_keyval_error:Nnn` 3, 87, 129
`\erw_keyval_keyonly:nn` 187
`\erw_keyval_parse:NNNn` 89, 146
`\erw_lambda:nnn` 3, 98
`\erw_option:n` 3, 174
`\erw_prop_keyval_parse:NNNn` . 4, 143
`\erw_prop_map_item:NNN` 4, 134
`\erw_prop_to_clist:Nn` ... 4, 121, 194
`\erw_seq_compose:nN` 4, 4, 197
`\erw_seq_compose_c:nN` 4, 201
`\erw_seq_compose_vers:nN` . 4, 205, 207
`\erw_seq_put_right_clist:Nn`
..... 4, 209, 213, 215
`\erw_seq_put_right_prop:NNn`
..... 4, 216, 220, 222
`\erw_seq_use:Nn` 4, 230
`\erw_sys_jobnametimestamp:` .. 4, 322
`\erw_sys_jobnametimestamp:nn` 4, 321
`\erw_sys_timestamp:` 4, 296, 331
`\erw_sys_timestamp:nn` ... 4, 290, 327
`\erw_sys_timestamp_delimiter:` 4, 323
`\erw_tl_append_item:nn` 5, 68, 397
`\erw_tl_compose:nN` 5, 401, 408
`\erw_tl_compose:nn` 5, 405
`\erw_tl_compose_c:nN` 5, 411, 418
`\erw_tl_compose_c:nn` 5, 415, 430
`\erw_tl_compose_vers:nN` .. 5, 421, 423
`\erw_tl_compose_vers:nn` 5, 425
`\erw_tl_fold:NN`
..... 5, 226, 403, 413, 434, 442
`\erw_tl_gset_function:N` 5, 443
`\erw_tl_gset_function:n` 5, 447
`\erw_tl_join:nn` . 5, 282, 288, 294, 459
`\erw_tl_join:nnn` 5, 266, 459
`\erw_tl_join:nnnn` 5, 459
`\erw_tl_join:nnnnn` 5, 459
`\erw_tl_last_item:n` 5, 451

`\erw_tl_map:n` ... 5, 158, 463, 470, 475
`\erw_tl_map:Nn` 5, 467
`\erw_tl_map_inline:nn` 5, 472
`\erw_tl_map_thread:Nn` 5, 517
`\erw_tl_map_thread_at:Nnn` 5, 496, 524
`\erw_tl_math_thread:Nn` 5
`\erw_tl_math_thread_at:Nnn` 5
`\erw_tl_repeat:nn` 5, 477
`\erw_tl_separators:n` 6, 234, 527
`\erw_tl_split:nn` 6, 492
`\erw_tl_split:nnn` 6, 481, 494

erw internal commands:

`__erw_cs_name:N` 4
`__erw_csint_ext_tl` 54
`\g_erw_csint_int` .. 19, 20, 32, 49, 53
`\g_erw_csint_name_tl` 20, 33
`__erw_function:n` 181, 186
`__erw_function:nn` 115, 119
`__erw_int_range:nnn` .. 56, 66, 79, 83
`__erw_keyval_function:n`
..... 91, 94, 123, 128
`__erw_keyval_function:nn` ... 92, 95
`__erw_lambda_expression` ... 101, 104
`__erw_oper_compose:NnN`
..... 152, 199, 203, 403, 413
`\g_erw_oper_fold_apply_par_tl` ..
..... 169, 439
`\g_erw_oper_fold_set_par_tl` 165, 436
`__erw_prop_map_item:NNN` ... 113, 137
`__erw_seq_fold:NN` . 199, 203, 223, 229
`\g_erw_seq_fold_item_tl`
..... 178, 225, 226, 227
`__erw_seq_put_right_clist:Nn` ...
..... 179, 190, 193, 212
`__erw_seq_put_right_prop:NNn` ...
..... 191, 196, 219
`__erw_sys_date:N` 236
`__erw_sys_date_dec:` 236, 278
`__erw_sys_date_hex:` 236, 279
`__erw_sys_datetime_base:n` . 256, 301
`__erw_sys_datetime_dec:` 278
`__erw_sys_datetime_dec:n` 256
`__erw_sys_datetime_hex:` 279
`__erw_sys_datetime_hex:n` 256
`__erw_sys_datetime_join:nn` ... 256
`__erw_sys_datetime_period:n` 256, 301
`__erw_sys_jobnametimestamp:` 286, 322
`__erw_sys_jobnametimestamp:n` .. 286
`__erw_sys_jobnametimestamp:nn` ..
..... 286, 321
`__erw_sys_jobnametimestamp_`
prefix: 280, 289, 295
`__erw_sys_set_delim:nn` 303, 313
`__erw_sys_time_dec:` 247, 278

__erw_sys_time_hex	247	keyval commands:	
__erw_sys_time_hex:	255, 279	\keyval_parse:NNn	93, 127, 185
__erw_sys_timestamp:nn	298, 329, 333		
\g__erw_sys_timestamp_delim_str .		M	
.....	266, 284, 306, 325	msg commands:	
\g__erw_tl_compose_tl		\msg_error:nnn	
....	335, 407, 408, 409, 417, 418, 419	207, 264, 276, 338, 423, 512
__erw_tl_map:nn	343, 465	\msg_error:nnnn	390
__erw_tl_map_thread_at:Nnn	348, 503	\msg_error:nnnnn	87
__erw_tl_map_thread_at:Nnnn	348, 504	\msg_error:nnnnnn	88
__erw_tl_map_thread_at:Nnnnn ..		\msg_new:nnn	
.....	348, 505	106, 107, 108, 109, 110, 111, 112
__erw_tl_map_thread_at:Nnnnnn ..			
.....	348, 506	O	
__erw_tl_separators:nn	374, 529	oper / fold_apply_par (option)	3
exp commands:		oper / fold_set_par (option)	3
\exp_args:Nf		options:	
....	124, 158, 351, 356, 357, 362,	oper / fold_apply_par	3
	363, 364, 369, 370, 371, 372, 498, 521	oper / fold_set_par	3
\exp_args:NNx	100	sys / timestamp_delim	3
\exp_args:No	300		
\exp_args:Nof	453	P	
\exp_args:Nx	68	prg commands:	
\exp_last_unbraced:Nf	6	\prg_replicate:nn	379
\exp_last_unbraced:NNf	232	prop commands:	
\exp_last_unbraced:No	312	\prop_if_exist:NTF	136, 145
\exp_last_unbraced:Nx	429	\prop_item:Nn	123
\exp_not:N	392	\prop_map_function:NN	119
\ExplSyntaxOff	531	\prop_new:N	139, 148
\ExplSyntaxOn	3		
		Q	
G		quark commands:	
g internal commands:		\quark_if_recursion_tail_stop:n	345
\g__erw_tl_function:n		\q_recursion_stop	465
....	154, 336, 346, 445, 449, 469, 474	\q_recursion_tail	465
I		S	
int commands:		seq commands:	
\int_case:nnTF	258, 376, 498	\seq_get_right:NN	225
\int_compare:nNnTF	58	\seq_if_exist:NTF	211, 218
\int_eval:n	60, 70, 73, 238, 249	\seq_new:N	213, 220
\int_incr:N	32	\seq_put_right:Nn	183, 227
\int_new:N	19	\seq_use:Nnnn	4, 233
\int_step_function:nnnN	28, 43	str commands:	
\int_step_inline:nn	85, 519	\str_case:nnTF	269
\int_step_inline:nnnn	479	\subsection	319
\int_to_alph:n	23, 25	sys / timestamp_delim (option)	3
\int_to_hex:n	245, 246, 255	sys commands:	
\int_zero:N	53	\c_sys_day_int	242
		\c_sys_hour_int	251
K		\c_sys_jobname_str	283
keys commands:		\c_sys_minute_int	252
\keys_define:nn	163, 308	\c_sys_month_int	241
\keys_set:nn	176	\c_sys_year_int	240

T			
tl commands:		<code>\tl_set:Nn</code>	20, 54, 407, 417
<code>\c_empty_tl</code>	263, 275, 388	<code>\tl_tail:n</code>	125, 486
<code>\tl_count:n</code>	456, 500, 521, 529	token commands:	
<code>\tl_head:n</code>	483, 521	<code>\token_if_cs:NTF</code>	35
<code>\tl_item:nn</code>	351, 356, 357,	U	
	362, 363, 364, 369, 370, 371, 372, 453	use commands:	
<code>\tl_map_function:nN</code>	428	<code>\use:N</code> .	23, 301, 305, 325, 436, 439, 484
<code>\tl_map_inline:nn</code>	484	<code>\use_i:nn</code>	383, 384
<code>\tl_new:N</code>	178, 335	<code>\use_i:nnn</code>	6
<code>\tl_range_braced:nnn</code>	44	<code>\use_ii:nn</code>	382, 384
<code>\tl_reverse:n</code>	160	<code>\usepackage</code>	2

Part IV

Implementation

1 Opening

```
1 <*package>
2 <@@=erw>
3 %      \ExplSyntaxOn
```

2 cs

2.1 backend

```
4 \cs_new:Nn \__erw_cs_name:N
5 {
6   \exp_last_unbraced:Nf \use_i:nnn {\cs_split_function:N #1}
7 }
```

2.2 frontend

`\erw_cs_identity:n`

```
8 \cs_set:Npn \erw_cs_identity:n #1{#1}
```

(End definition for `\erw_cs_identity:n`. This function is documented on page 4.)

`\erw_cs_set_inline:Nn`

`\erw_cs_gset_inline:Nn`

```
9 \cs_new:Nn \erw_cs_set_inline:Nn
10 {
11   \cs_set:Npn #1 ##1{#2}
12 }
13 \cs_generate_variant:Nn \erw_cs_set_inline:Nn {cn}
```

(End definition for `\erw_cs_set_inline:Nn` and `\erw_cs_gset_inline:Nn`. These functions are documented on page 4.)

`\erw_cs_gset_inline:Nn`

```
14 \cs_new:Nn \erw_cs_gset_inline:Nn
15 {
16   \cs_gset:Npn #1 ##1{#2}
17 }
18 \cs_generate_variant:Nn \erw_cs_gset_inline:Nn {cn}
```

(End definition for `\erw_cs_gset_inline:Nn`. This function is documented on page 4.)

3 csint

3.1 backend

```
19 \int_new:N \g__erw_csint_int
20 \tl_set:Nn \g__erw_csint_name_tl {\erw_csint_name:n{\g__erw_csint_int}}
```

3.2 frontend

`\erw_csint:nn`

```

21 \cs_new:Nn \erw_csint:nn
22 {
23   \use:c{__erw_csint_\int_to_alph:n{#1}:n}{#2}
24 }

```

(End definition for `\erw_csint:nn`. This function is documented on page 4.)

`\erw_csint_name:n`

```

25 \cs_new:Nn \erw_csint_name:n {__erw_csint_\int_to_alph:n{#1}:n}

```

(End definition for `\erw_csint_name:n`. This function is documented on page 4.)

`\erw_csint_names:nnn`

```

26 \cs_new:Nn \erw_csint_names:nnn
27 {
28   \int_step_function:nnnN { #1 }{ #2 }{ #3 } \erw_csint_name:n
29 }

```

(End definition for `\erw_csint_names:nnn`. This function is documented on page 4.)

`\erw_csint_new:n`

```

30 \cs_new_protected:Nn \erw_csint_new:n
31 {
32   \int_incr:N \g__erw_csint_int
33   \erw_cs_set_inline:cn{\g__erw_csint_name_tl}
34   {
35     \token_if_cs:NTF
36     {#1}
37     {#1{##1}}
38     {#1}
39   }
40 }

```

(End definition for `\erw_csint_new:n`. This function is documented on page 4.)

`\erw_csint_names_braced:nnn`

`\erw_csint_names_braced:n`
`\erw_csint_names_braced:`

```

41 \cs_new:Nn \erw_csint_names_braced:nnn
42 {
43   \int_step_function:nnnN { #1 }{ #2 }{ #3 } \erw_csint_names_braced:n
44   % TODO \tl_range_braced:nnn?
45 }
46 \cs_new:Nn \erw_csint_names_braced:n {{\erw_csint_name:n{#1}}}
47 \cs_new:Nn \erw_csint_names_braced:
48 {
49   \erw_csint_names_braced:nnn{1}{1}{\g__erw_csint_int}
50 }

```

(End definition for `\erw_csint_names_braced:nnn`, `\erw_csint_names_braced:n`, and `\erw_csint_names_braced:`. These functions are documented on page 4.)

`\erw_csint_reset:`

```

51 \cs_new_protected:Nn \erw_csint_reset:
52 {
53   \int_zero:N \g__erw_csint_int
54   \tl_set:Nn \__erw_csint_ext_tl{}}%^^A TODO remove?
55 }

```

(End definition for `\erw_csint_reset:`. This function is documented on page 4.)

4 int

4.1 backend

```

56 \cs_set:Npn \__erw_int_range:nnn #1 #2 #3
57 {
58   \int_compare:nNnTF
59   {
60     \int_eval:n{#2+1}
61   }>{#3}
62   {
63     {#1}
64   }
65   {
66     \__erw_int_range:nnn
67     {
68       \exp_args:Nx\erw_tl_append_item:nn{#1}
69       {
70         \int_eval:n{#2+1}
71       }
72     }
73     {\int_eval:n{#2+1}}
74     {#3}
75   }
76 }

```

4.2 frontend

`\erw_int_range:nn`

`\erw_int_range:n`

```

77 \cs_new:Nn \erw_int_range:nn
78 {
79   \__erw_int_range:nnn {{#1}}{#1}{#2}
80 }
81 \cs_new:Nn \erw_int_range:n
82 {
83   \__erw_int_range:nnn {}{0}{#1}
84   % ^^A Alt to:
85   % ^^A   \int_step_inline:nn {#1}{##1}
86 }

```

(End definition for `\erw_int_range:nn` and `\erw_int_range:n`. These functions are documented on page 5.)

5 keys

5.1 frontend

```
\erw_keyval_error:Nn
\erw_keyval_error:Nnn
```

```
87 \cs_new:Nn \erw_keyval_error:Nn{\msg_error:nnnnn{__erw}{keyval/n}{\erw_keyval_error:Nn}{#1}{#2}}
88 \cs_new:Nn \erw_keyval_error:Nnn{\msg_error:nnnnnn{__erw}{keyval/nn}{\erw_keyval_error:Nnn}{#1}{#2}{#3}}
```

(End definition for `\erw_keyval_error:Nn` and `\erw_keyval_error:Nnn`. These functions are documented on page 5.)

```
\erw_keyval_parse:NNNn
```

```
89 \cs_new_protected:Nn\erw_keyval_parse:NNNn
90 {
91   \cs_set_protected:Nn \__erw_keyval_function:n {#2 #1{##1}}
92   \cs_set_protected:Nn \__erw_keyval_function:nn {#3 #1{##1}{##2}}
93   \keyval_parse:NNn
94   \__erw_keyval_function:n
95   \__erw_keyval_function:nn
96   {#4}
97 }
```

(End definition for `\erw_keyval_parse:NNNn`. This function is documented on page ??.)

6 lambda

```
\erw_lambda:nnn
```

```
98 \cs_new_protected:Npn \erw_lambda:nnn #1 #2 #3
99 {
100   \exp_args:NNx
101   #1 \__erw_lambda_expression
102   {#2}
103   {#3}
104   \__erw_lambda_expression
105 }
```

(End definition for `\erw_lambda:nnn`. This function is documented on page 5.)

7 msg

7.1 backend

```
106 \msg_new:nnn{__erw}{generic}{#1}
107 \msg_new:nnn{__erw}{keyval/nn}{#1#2{#3}{#4}};~encountered~key=val~where~only~key~required}
108 \msg_new:nnn{__erw}{keyval/n}{#1#2{#3}};~encountered~key~~where~only~key=val~required}
109 \msg_new:nnn{__erw}{separ}{#1~expects~1~to~3~items,~#2}
110 \msg_new:nnn{__erw}{timestamp / base}{Calling~#1,~arg~must~be~'dec|hex'}
111 \msg_new:nnn{__erw}{timestamp / period}{Calling~#1,~arg~must~be~'date|time|datetime'}
```

7.2 frontend

```
112 \msg_new:nnn{erw}{notset}{#1~not~set}
```

8 prop

8.1 backend

```
113 \cs_new_protected:Nn \__erw_prop_map_item:NNN
114 {
115   \cs_set_protected:Nn \__erw_function:nn
116   {
117     #1 #2 {##1}{##2}
118   }
119   \prop_map_function:NN #3 \__erw_function:nn
120 }
```

8.2 frontend

\erw_prop_to_clist:Nn

```
121 \cs_new_protected:Nn \erw_prop_to_clist:Nn
122 {
123   \cs_set:Nn \__erw_keyval_function:n {,\prop_item:Nn#1{##1}}
124   \exp_args:Nf
125   \tl_tail:n
126   {
127     \keyval_parse:NNn
128     \__erw_keyval_function:n
129     \erw_keyval_error:Nnn
130     {#2}
131   }
132 }
133 \cs_generate_variant:Nn \erw_prop_to_clist:Nn { c }
```

(End definition for \erw_prop_to_clist:Nn. This function is documented on page 5.)

\erw_prop_map_item:NNN

```
134 \cs_new_protected:Nn \erw_prop_map_item:NNN
135 {
136   \prop_if_exist:NTF #2
137   {\__erw_prop_map_item:NNN #1#2#3}
138   {
139     \prop_new:N #2
140     \erw_prop_map_item:NNN #1#2#3
141   }
142 }
```

(End definition for \erw_prop_map_item:NNN. This function is documented on page 5.)

\erw_prop_keyval_parse:NNNn

```
143 \cs_new_protected:Nn \erw_prop_keyval_parse:NNNn
144 {
145   \prop_if_exist:NTF#1
146   {\erw_keyval_parse:NNNn #1#2#3{#4}}
147   {
148     \prop_new:N #1
149     \erw_prop_keyval_parse:NNNn#1#2#3{#4}
150   }
151 }
```

(End definition for \erw_prop_keyval_parse:NNNn. This function is documented on page 5.)

9 oper

9.1 backend

```
152 \cs_new:Nn \__erw_oper_compose:NnN
153 {
154   \erw_cs_set_inline:Nn \g__erw_tl_function:n
155   {
156     #1{##1}#3
157   }
158   \exp_args:Nf\erw_tl_map:n
159   {
160     \tl_reverse:n{#2}
161   }
162 }
```

9.2 frontend

```
163 \keys_define:nn{__erw}
164 {
165   oper/fold_set_par.tl_gset:N = \g__erw_oper_fold_set_par_tl,
166   oper/fold_set_par.value_required:n = true,
167   oper/fold_set_par.default:n = {Nf},
168   oper/fold_set_par.initial:n = {Nf},
169   oper/fold_apply_par.tl_gset:N = \g__erw_oper_fold_apply_par_tl,
170   oper/fold_apply_par.value_required:n = true,
171   oper/fold_apply_par.default:n = {Nf},
172   oper/fold_apply_par.initial:n = {Nf}
173 }
```

10 option

```
174 \cs_new_protected:Nn\erw_option:n
175 {
176   \keys_set:nn{__erw}{#1}
177 }
```

11 seq

11.1 backend

```
178 \tl_new:N \g__erw_seq_fold_item_tl
179 \cs_new_protected:Nn\__erw_seq_put_right_clist:Nn
180 {
181   \cs_set_protected:Nn \__erw_function:n
182   {
183     \seq_put_right:Nn #1{##1}
184   }
185   \keyval_parse:NNn
186   \__erw_function:n
187   \erw_keyval_keyonly:nn
188   {#2}
189 }
190 \cs_generate_variant:Nn \__erw_seq_put_right_clist:Nn { c }
191 \cs_new_protected:Nn\__erw_seq_put_right_prop:NNn
```

```

192 {
193   \__erw_seq_put_right_clist:Nn #1
194   {\erw_prop_to_clist:Nn #2 {#3}}
195 }
196 \cs_generate_variant:Nn \__erw_seq_put_right_prop:NNn { cc }

```

11.2 frontend

```

197 \cs_new:Nn \erw_seq_compose:nN
198 {
199   \__erw_oper_compose:NnN \__erw_seq_fold:NN {#1} #2
200 }
201 \cs_new:Nn \erw_seq_compose_c:nN
202 {
203   \__erw_oper_compose:NnN \__erw_seq_fold:cN {#1} #2
204 }
205 \cs_new:Nn \erw_seq_compose_vers:nN
206 {
207   \msg_error:nnn{\__erw}{notset}{\erw_seq_compose_vers:nN}
208 }
209 \cs_new_protected:Nn \erw_seq_put_right_clist:Nn
210 {
211   \seq_if_exist:NTF#1
212   {\__erw_seq_put_right_clist:Nn#1{#2}}
213   {\seq_new:N#1\erw_seq_put_right_clist:Nn#1{#2}}
214 }
215 \cs_generate_variant:Nn \erw_seq_put_right_clist:Nn { c }
216 \cs_new_protected:Nn \erw_seq_put_right_prop:NNn
217 {
218   \seq_if_exist:NTF#1
219   {\__erw_seq_put_right_prop:NNn#1#2{#3}}
220   {\seq_new:N#1\erw_seq_put_right_prop:NNn#1#2{#3}}
221 }
222 \cs_generate_variant:Nn \erw_seq_put_right_prop:NNn { cc }
223 \cs_new:Nn \__erw_seq_fold:NN
224 {
225   \seq_get_right:NN #2 \g__erw_seq_fold_item_tl
226   \erw_tl_fold:NN #1 \g__erw_seq_fold_item_tl
227   \seq_put_right:No #2 {\g__erw_seq_fold_item_tl}
228 }
229 \cs_generate_variant:Nn \__erw_seq_fold:NN { cN }
230 \cs_new:Nn \erw_seq_use:Nn
231 {
232   \exp_last_unbraced:NNf
233   \seq_use:Nnnn #1
234   \erw_tl_separators:n{#2}
235 }

```

12 sys

12.1 backend

```

\__erw_sys_date:N
\__erw_sys_date_dec:
\__erw_sys_date_hex:
236 \cs_new:Nn \__erw_sys_date_dec:

```

```

237 {
238   \int_eval:n
239   {
240     \c_sys_year_int * 10000
241     +\c_sys_month_int * 100
242     +\c_sys_day_int * 1
243   }
244 }
245 \cs_new:Nn \__erw_sys_date:N{\int_to_hex:n{\__erw_sys_date_dec:}}
246 \cs_new:Nn \__erw_sys_date_hex:{\int_to_hex:n{\__erw_sys_date_dec:}}

(End definition for \__erw_sys_date:N, \__erw_sys_date_dec:, and \__erw_sys_date_hex:.)

```

```

\__erw_sys_time_dec:
\__erw_sys_time_hex
247 \cs_new:Nn \__erw_sys_time_dec:
248 {
249   \int_eval:n
250   {
251     \c_sys_hour_int * 100
252     +\c_sys_minute_int * 1
253   }
254 }
255 \cs_new:Nn \__erw_sys_time_hex:{\int_to_hex:n{\__erw_sys_time_dec:}}

(End definition for \__erw_sys_time_dec: and \__erw_sys_time_hex.)

```

```

\__erw_sys_datetime_base:n
\__erw_sys_datetime_dec:n
\__erw_sys_datetime_join:nn
\__erw_sys_datetime_hex:n
\__erw_sys_datetime_period:n
256 \cs_new:Nn \__erw_sys_datetime_base:n
257 {
258   \int_case:nnTF{#1}
259   {
260     {10}{dec}
261     {16}{hex}
262   }
263   {\c_empty_tl}
264   {\msg_error:nnn{\__erw}{timestamp / base}{\__erw_sys_datetime_base:n{#1}}}
265 }
266 \cs_new:Nn \__erw_sys_datetime_join:nn{\erw_tl_join:nnn{#1}{\g__erw_sys_timestamp_delim_str}{#2}}
267 \cs_new:Nn \__erw_sys_datetime_period:n
268 {
269   \str_case:nnTF{#1}
270   {
271     {date}{date}
272     {time}{time}
273     {datetime}{datetime}
274   }
275   {\c_empty_tl}
276   {\msg_error:nnn{\__erw}{timestamp / period}{\__erw_sys_datetime_period:n{#1}}}
277 }
278 \cs_new:Nn \__erw_sys_datetime_dec: {\__erw_sys_datetime_join:nn{\__erw_sys_date_dec:}{\__erw_sys_time_dec:}}
279 \cs_new:Nn \__erw_sys_datetime_hex: {\__erw_sys_datetime_join:nn{\__erw_sys_date_hex:}{\__erw_sys_time_hex:}}

(End definition for \__erw_sys_datetime_base:n and others.)

```

_erw_sys_jobnametimestamp_prefix:

```

280 \cs_new:Nn\_erw_sys_jobnametimestamp_prefix:
281 {
282   \erw_tl_join:nn
283   {\c_sys_jobname_str}
284   {\g__erw_sys_timestamp_delim_str}
285 }

```

(End definition for _erw_sys_jobnametimestamp_prefix:.)

_erw_sys_jobnametimestamp:n

_erw_sys_jobnametimestamp:

```

286 \cs_new:Nn\_erw_sys_jobnametimestamp:nn
287 {
288   \erw_tl_join:nn
289   {\_erw_sys_jobnametimestamp_prefix:}
290   {\erw_sys_timestamp:nn{#1}{#2}}
291 }
292 \cs_new:Nn\_erw_sys_jobnametimestamp:
293 {
294   \erw_tl_join:nn
295   {\_erw_sys_jobnametimestamp_prefix:}
296   {\erw_sys_timestamp:}
297 }

```

(End definition for _erw_sys_jobnametimestamp:n and _erw_sys_jobnametimestamp:.)

_erw_sys_timestamp:nn

```

298 \cs_new:Nn\_erw_sys_timestamp:nn
299 {
300   \exp_args:No
301   \use:c{__erw_sys\_erw_sys_datetime_period:n{#1}\_erw_sys_datetime_base:n{#2}:}
302 }
303 \cs_new_protected:Nn \_erw_sys_set_delim:nn
304 {
305   \use:c{tl_gset:N#1}
306   \g__erw_sys_timestamp_delim_str{#2}
307 }

```

(End definition for _erw_sys_timestamp:nn.)

```

308 \keys_define:nn{__erw}
309 {
310   sys / timestamp_delim .code:n =
311   {
312     \exp_last_unbraced:No
313     \_erw_sys_set_delim:nn{n}{#1}
314   },
315   sys / timestamp_delim .value_required:n = true,
316   sys / timestamp_delim .default:n = {-},
317   sys / timestamp_delim .initial:n = {-}
318 }
319 % \subsection{frontend}
320 % \begin{macrocode}
321 \cs_new:Nn\erw_sys_jobnametimestamp:nn{\_erw_sys_jobnametimestamp:nn{#1}{#2}}
322 \cs_new:Nn\erw_sys_jobnametimestamp:{\_erw_sys_jobnametimestamp:}

```

```

323 \cs_new:Nn\erw_sys_timestamp_delimiter:
324 {
325   \use:N \g__erw_sys_timestamp_delim_str
326 }
327 \cs_new:Nn\erw_sys_timestamp:nn
328 {
329   \__erw_sys_timestamp:nn{#1}{#2}
330 }
331 \cs_new:Nn\erw_sys_timestamp:
332 {
333   \__erw_sys_timestamp:nn{datetime}{16}
334 }

```

13 tl

13.1 backend

```

335 \tl_new:N \g__erw_tl_compose_tl

\g__erw_tl_function:n

336 \cs_new_protected:Nn \g__erw_tl_function:n
337 {
338   \msg_error:nnn
339   {erw}
340   {notset}
341   {\g__erw_tl_function:n}
342 }

(End definition for \g__erw_tl_function:n.)

\__erw_tl_map:nn

343 \cs_set_protected:Nn \__erw_tl_map:nn
344 {
345   \quark_if_recursion_tail_stop:n{#1}
346   \g__erw_tl_function:n{#1} \__erw_tl_map:nn{#2}
347 }

(End definition for \__erw_tl_map:nn.)

\__erw_tl_map_thread_at:Nnn
\__erw_tl_map_thread_at:Nnnn
  \__erw_tl_map_thread_at:Nnnnn
    \__erw_tl_map_thread_at:Nnnnnn

348 \cs_set_protected:Nn \__erw_tl_map_thread_at:Nnn
349 {
350   #1
351   {\exp_args:Nf\tl_item:nn {#3} {#2} }
352 }
353 \cs_set_protected:Nn \__erw_tl_map_thread_at:Nnnn
354 {
355   #1
356   {\exp_args:Nf\tl_item:nn {#3} {#2} }
357   {\exp_args:Nf\tl_item:nn {#4} {#2} }
358 }
359 \cs_set_protected:Nn \__erw_tl_map_thread_at:Nnnnn
360 {
361   #1

```

```

362   {\exp_args:Nf\tl_item:nn {#3} {#2} }
363   {\exp_args:Nf\tl_item:nn {#4} {#2} }
364   {\exp_args:Nf\tl_item:nn {#5} {#2} }
365 }
366 \cs_set_protected:Nn \__erw_tl_map_thread_at:Nnnnnn
367 {
368   #1
369   {\exp_args:Nf\tl_item:nn {#3} {#2} }
370   {\exp_args:Nf\tl_item:nn {#4} {#2} }
371   {\exp_args:Nf\tl_item:nn {#5} {#2} }
372   {\exp_args:Nf\tl_item:nn {#6} {#2} }
373 }

```

(End definition for __erw_tl_map_thread_at:Nnn and others.)

```

\__erw_tl_separators:nn #1 : < int >
                        #2 : < items >

374 \cs_new:Nn \__erw_tl_separators:nn
375 {
376   \int_case:nnTF {#1}
377   {
378     {1}
379     { \prg_replicate:nn{ 3 }{#2} }
380     {2}
381     {
382       { \use_ii:nn #2 }
383       { \use_i:nn #2 }
384       { \use_i:nn #2 \use_ii:nn #2 }
385     }
386     {3}{#2}
387   }
388   { \c_empty_tl }
389   {
390     \msg_error:nnnn { __erw }
391     { separ }
392     { \exp_not:N \__erw_tl_separators:nn }
393     {#2}
394   }
395 }
396 \cs_generate_variant:Nn \__erw_tl_separators:nn { e }

```

(End definition for __erw_tl_separators:nn.)

13.2 frontend

```

397 \cs_new:Nn \erw_tl_append_item:nn
398 {
399   {#1{#2}}
400 }
401 \cs_new:Nn \erw_tl_compose:nN
402 {
403   \__erw_oper_compose:NnN \erw_tl_fold:NN {#1} #2
404 }
405 \cs_new:Nn \erw_tl_compose:nn

```



```

406 {
407   \tl_set:Nn \g__erw_tl_compose_tl {#2}
408   \erw_tl_compose:nN{#1}\g__erw_tl_compose_tl
409   \g__erw_tl_compose_tl
410 }
411 \cs_new:Nn \erw_tl_compose_c:nN
412 {
413   \__erw_oper_compose:NnN \erw_tl_fold:cN {#1} #2
414 }
415 \cs_new:Nn \erw_tl_compose_c:nn
416 {
417   \tl_set:Nn \g__erw_tl_compose_tl {#2}
418   \erw_tl_compose_c:nN{#1}\g__erw_tl_compose_tl
419   \g__erw_tl_compose_tl
420 }
421 \cs_new:Nn \erw_tl_compose_vers:nN
422 {
423   \msg_error:nnn{__erw}{notset}{\erw_tl_compose_vers:nN}
424 }
425 \cs_new:Nn \erw_tl_compose_vers:nn
426 {
427   \erw_csint_reset:{}
428   \tl_map_function:nN{#1}\erw_csint_new:n
429   \exp_last_unbraced:Nx
430   \erw_tl_compose_c:nn
431   {{\erw_csint_names_braced:{}}}
432   {#2}
433 }
434 \cs_new:Nn \erw_tl_fold:NN
435 {
436   \use:c{tl_set:\g__erw_oper_fold_set_par_tl}
437   #2
438   {
439     \use:c{exp_args:\g__erw_oper_fold_apply_par_tl}{#1}{#2}
440   }
441 }
442 \cs_generate_variant:Nn \erw_tl_fold:NN {cN}
443 \cs_new:Nn \erw_tl_gset_function:N
444 {
445   \erw_cs_gset_eq:NN \g__erw_tl_function:n #1
446 }
447 \cs_new:Nn \erw_tl_gset_function:n
448 {
449   \erw_cs_gset_inline:Nn \g__erw_tl_function:n {#1}
450 }
451 \cs_new:Nn \erw_tl_last_item:n
452 {
453   \exp_args:Nof \tl_item:nn
454   {#1}
455   {
456     \tl_count:n{#1}
457   }
458 }

```

```

\erw_tl_join:nn
\erw_tl_join:nnn
\erw_tl_join:nnnn
\erw_tl_join:nnnnn

```

```

459 \cs_new:Nn \erw_tl_join:nn{#1#2}
460 \cs_new:Nn \erw_tl_join:nnn{#1#2#3}
461 \cs_new:Nn \erw_tl_join:nnnn{#1#2#3#4}
462 \cs_new:Nn \erw_tl_join:nnnnn{#1#2#3#4#5}

```

(End definition for \erw_tl_join:nn and others. These functions are documented on page 7.)

```

463 \cs_new_protected:Nn \erw_tl_map:n
464 {
465   \__erw_tl_map:nn#1\q_recursion_tail\q_recursion_stop\q_recursion_tail\q_recursion_stop
466 }
467 \cs_new_protected:Nn \erw_tl_map:Nn
468 {
469   \cs_set_eq:NN \g__erw_tl_function:n #1
470   \erw_tl_map:n{#2}
471 }
472 \cs_new_protected:Nn \erw_tl_map_inline:nn
473 {
474   \erw_cs_set_inline:Nn \g__erw_tl_function:n {#1}
475   \erw_tl_map:n{#2}
476 }
477 \cs_new:Nn \erw_tl_repeat:nn
478 {
479   \int_step_inline:nnnn{1}{1}{#1}{#2}
480 }
481 \cs_new:Nn \erw_tl_split:nnn
482 {
483   \tl_head:n{#1}
484   \use:c{exp_args:#3} \tl_map_inline:nn
485   {
486     \tl_tail:n
487     {
488       #1
489     }
490   }{#2##1}
491 }
492 \cs_new:Nn \erw_tl_split:nn
493 {
494   \erw_tl_split:nnn{#1}{#2}{Nf}
495 }
496 \cs_new_protected:Nn \erw_tl_map_thread_at:Nnn
497 {
498   \exp_args:Nf\int_case:nnTF
499   {
500     \tl_count:n{#3}
501   }
502   {
503     {1}{ \__erw_tl_map_thread_at:Nnn #1{#2}#3 }
504     {2}{ \__erw_tl_map_thread_at:Nnnn #1{#2}#3 }
505     {3}{ \__erw_tl_map_thread_at:Nnnnn #1{#2}#3 }
506     {4}{ \__erw_tl_map_thread_at:Nnnnnn #1{#2}#3 }
507   }
508   {
509     % Do nothing

```

```

510   }
511   {
512     \msg_error:nnn{__erw}
513     {generic}
514     {erw_tl_map_thread_at:~count~of~#3~not~withing~1~to~4}
515   }
516 }
517 \cs_new_protected:Nn \erw_tl_map_thread:Nn
518 {
519   \int_step_inline:nn
520   {
521     \exp_args:Nf \tl_count:n{ \tl_head:n{#2} }
522   }
523   {
524     \erw_tl_map_thread_at:Nnn #1 {##1} {#2}
525   }
526 }
527 \cs_new:Nn \erw_tl_separators:n
528 {
529   \__erw_tl_separators:en{ \tl_count:n{#1} }{#1}
530 }

```

14 Closing

```

531 \ExplSyntaxOff
532 \endpackage

```