

# The ccool package\*

Erwann Rogard†

Released 2020/04/12

## Abstract

ccool stands for Custom COntent Oriented for L<sup>A</sup>T<sub>E</sub>X, a concept pioneered by cool[1]<sup>1</sup>. This is done using a minimalist interface built upon xparse[4]. Specifically, `\Ccool<name>` begins a series of instructions alternating between ‘text’ and macro definitions, that themselves optionally expand using predefined or inline rules. For example,

```
\Ccool<Math>[Let~]
i{\mathbb{#1}}{ Nat = N, Real = R }*s{{~\rm{and}}~}}
[~denote the natural and real numbers.]{}
```

expands to: “Let  $\mathbb{N}$  and  $\mathbb{R}$  denote the natural and real numbers.” As a side effect, `$_\text{Nat}<Math>$` encodes “N” (and likewise for `\Real`). `Math` being the default for `<name>`, `<Math>` can be dropped. Optionally, the macros can be written to a file, and read, which can be useful for typesetting documents sharing the same notation.

## Contents

<b>I</b>	<b>Usage</b>	<b>3</b>
<b>0</b>	<b>Convention</b>	<b>3</b>
<b>1</b>	<b>Loading the package</b>	<b>4</b>
<b>2</b>	<b>\Ccool</b>	<b>4</b>
2.1	<code>&lt;tl<sub>1</sub>&gt;</code> . . . . .	4
2.2	<code>[tl<sub>2</sub>]</code> . . . . .	4
2.3	<code>i{&lt;code<sub>1</sub>&gt;}</code> . . . . .	4
2.4	<code>{&lt;kv<sub>1</sub>&gt;}</code> . . . . .	4
2.5	<code>+</code> . . . . .	4
2.6	<code>*</code> . . . . .	5
2.7	<code>s{{&lt;tl<sub>3</sub>&gt;}{&lt;tl<sub>3</sub>&gt;}{&lt;tl<sub>4</sub>&gt;}{&lt;tl<sub>3</sub>&gt;}{&lt;tl<sub>4</sub>&gt;}{&lt;tl<sub>5</sub>&gt;}}</code> . . . . .	5
2.8	<code>o{&lt;code<sub>2</sub>&gt;}</code> . . . . .	5
2.9	<code>[tl<sub>6</sub>]</code> . . . . .	5
<b>3</b>	<b>\CcoolClear</b>	<b>5</b>

---

\*This file describes version v1.8, last revised 2020/04/12.

†firstname dot lastname AusTria gmail dot com

<sup>1</sup>Whereas cool provided predefined macros, ccool is tool for making macros, hence “custom”.

4	\CcoolHook	5
5	\CcoolLambda	5
6	\CcoolOption	5
6.1	Expans . . . . .	5
6.2	File . . . . .	6
6.3	Inner . . . . .	6
6.4	Name . . . . .	6
6.5	Outer . . . . .	6
6.6	Separ . . . . .	6
6.7	Write . . . . .	6
7	\CcoolRead	6
8	\CcoolVers	6
9	Do's and dont's	6
 II Listing		8
Listing 1. Version.		8
Listing 2. Preamble.		8
Listing 3. Separators.		8
Listing 4. Hello, world!		8
Listing 5. Listing 4 read from file.		9
Listing 6. Probability space.		9
Listing 7. Listing 6 read from file.		9
Listing 8. Mittelwertsatz für $n$ Variable.		10
Listing 9. Listing 8 read from file.		10
Listing 10. Lambda expression.		10
Listing 11. Listing 10 read from file.		11
Listing 12. CUSUM statistic.		11
Listing 13. Listing 12 read from file.		12
 III Other		13
1	Acknowledgment	13

<b>2</b>	<b>Install</b>	<b>13</b>
<b>3</b>	<b>Issue</b>	<b>13</b>
<b>4</b>	<b>Support</b>	<b>13</b>
<b>5</b>	<b>Testing</b>	<b>13</b>
5.1	Technicality . . . . .	13
5.2	Platform . . . . .	13
5.3	Engine . . . . .	13
5.4	Results . . . . .	13
5.5	Other . . . . .	14
<b>IV</b>	<b>Implementation</b>	<b>15</b>
<b>1</b>	<b>aux</b>	<b>15</b>
<b>2</b>	<b>lambda</b>	<b>17</b>
<b>3</b>	<b>log</b>	<b>18</b>
<b>4</b>	<b>make_key</b>	<b>19</b>
<b>5</b>	<b>make_ccool</b>	<b>19</b>
<b>6</b>	<b>msg</b>	<b>21</b>
<b>7</b>	<b>option</b>	<b>21</b>
<b>8</b>	<b>prop</b>	<b>22</b>
<b>9</b>	<b>seq</b>	<b>24</b>
<b>10</b>	<b>Front-end</b>	<b>24</b>
<b>11</b>	<b>Misc</b>	<b>26</b>

## Part I

# Usage

## Convention

1. Loosely, those of [2] and [4], for example as to the meaning of  $\langle token\ list \rangle$ .
2. If unspecified, the environment in which a macro must be declared is `document`.

---

<code>\usepackage</code>	<code>\usepackage{ccool}</code>
--------------------------	---------------------------------

---

### Requirement

1. `ccool.sty` is in the path of the L<sup>A</sup>T<sub>E</sub>X engine. See [Part III, section 4](#).
2. Declare it in the *preamble*

---

<code>\Ccool</code>	<code>\Ccool&lt;tl<sub>1</sub>&gt;</code> <code>[\tl<sub>2</sub>]</code> <code>if{code<sub>1</sub>}</code> <code>{kv<sub>1</sub>}</code> <code>+</code> <code>*</code> <code>s{\tl<sub>3</sub>}\tl<sub>3</sub>\tl<sub>4</sub>\tl<sub>3</sub>\tl<sub>4</sub>\tl<sub>5</sub>}</code> <code>of{code<sub>2</sub>}</code> <code>[\tl<sub>6</sub>]</code>
---------------------	---

---

**Requirement** `kv1` is specified (all others optional).

`<tl1>`

**Example** `Math, ModelA, ModelB`

**Semantics** Identifies a group of macros

`<tl2>`

**Example** `Let~`

**Semantics** Expands `<tl2>`

`<code1>`

**Example** `\mathbb{#1}`

**Semantics**

1.  $\langle val_i \rangle \leftarrow \langle code_1 \rangle$  applied to  $\langle val_i \rangle$

`<kv1>`

**Example** `Elms={\omega_1, \dots, \omega_n}, Sample=\Omega`

**Semantics**

2.  $\langle key_i \rangle \langle tl_1 \rangle \leftarrow \langle val_i \rangle$  defined in step 1, using **Expans** for expansion.
3. If **Write**, writes the input used by step 2 to **File**

+

**Other** Needed to make `\Ccool`'s side effect within a *local group* persist thereafter

**Semantics** Appends step 2 and step 3 to `\CcoolHook`

\*

### Semantics

4. Expands  $\langle code_2 \rangle$  applied to the list created in step 1, using the separator specified by  $\langle tl_3 \rangle$ ,  $\langle tl_4 \rangle$ ,  $\langle tl_5 \rangle$ .

$\langle tl_3 \rangle$

**Example**  $\{\sim\backslash in\sim\}$

$\langle tl_4 \rangle$

**Example**  $\{,\sim\}$

$\langle tl_5 \rangle$

**Example**  $\{\sim\backslash \&\sim\}$

$\langle code_2 \rangle$

**Example**  $\$\left\{\#1\right\}\right\right\right\right\}$

$\langle tl_6 \rangle$

**Semantics**  $\backslash Ccool<\langle tl_1 \rangle>[<\langle tl_6 \rangle]$

---

---

$\backslash CcoolClear$   $\backslash CcoolClear<\langle keyval list \rangle>$

**Semantics** Clears any data created by  $\backslash Ccool\{\langle tl_1 \rangle\}$ , for all  $\langle tl_1 \rangle$  in  $\langle keyval list \rangle$

---

---

$\backslash CcoolHook$   $\backslash CcoolHook$

**Example**  $\backslash AfterEndEnvironment\{theorem\}\{\backslash CcoolHook\}$

---

---

$\backslash CcoolLambda$   $\backslash CcoolLambda[<\langle integer \rangle]\{\langle code \rangle\}$

**Example**  $\backslash Ccool\{ EvalAt = \backslash CcoolLambda\{(\#1)\} \}$

**Semantics** Creates a lambda expression with  $\langle integer \rangle$  arguments for  $\langle code \rangle$

---

---

$\backslash CcoolOption$   $\backslash CcoolOption\{\langle kv10 \rangle\}$

**Semantics** Set default options for  $\backslash Ccool$

Expans

	<p><b>Default</b> <code>xo</code></p> <p><b>Syntax</b> Either of <code>eo</code>, <code>ee</code>, <code>ex</code>, <code>xe</code>, <code>xo</code>, <code>xe</code>, <code>xx</code></p>
File	<p><b>Default</b> <code>ccool\pdfcreationdate</code></p> <p><b>Syntax</b> Expands to a valid <i>path</i></p>
Inner	<p><b>Default</b> <code>####1</code></p> <p><b>Semantics</b> Default for <math>\langle code_1 \rangle</math></p> <p><b>Syntax</b> Use <code>####1</code> as the argument to be replaced</p>
Name	<p><b>Default</b> <code>Math</code></p> <p><b>Semantics</b> Default for <math>\langle tl_1 \rangle</math></p>
Outer	<p><b>Default</b> <code>\ensuremath{####1}</code></p> <p><b>Semantics</b> Default for <math>\langle code_2 \rangle</math></p> <p><b>Syntax</b> Use <code>####1</code> as the argument to be replaced</p>
Separ	<p><b>Default</b> <code>{ {\ }and{\ } } { ,{\ } } { ,{\ }and{\ } }</code></p> <p><b>Semantics</b> Default for <code>separators'</code> parameter</p> <p><b>Syntax</b> That of ‘separators’ in [2, Section 8 of l3seq]</p>
Write	<p><b>Default</b> <code>\BooleanFalse</code></p> <p><b>Syntax</b> <i>Boolean</i></p>
<hr/> <hr/>	
<u><code>\CcoolRead</code></u>	<p><code>\CcoolRead[\langle path \rangle]</code></p> <p><b>Other</b> The default for <math>\langle path \rangle</math> is the last write-file (see <math>\langle kv_1 \rangle</math>)</p> <p><b>Semantics</b></p> <ol style="list-style-type: none"> <li>1. Reads the definitions in <math>\langle path \rangle</math>.</li> <li>2. Writes to <code>ccool.log</code>: ‘read from <math>\langle path \rangle</math>’</li> </ol>
<hr/> <hr/>	
<u><code>\CcoolVers</code></u>	<p><code>\CcoolVers</code></p> <p><b>Semantics</b> Expands to the package’s version</p>

## Do's and don't's

1.

Don't: `\Ccool{ A = a, B = b }[Hello, world!]`.

Do: `\Ccool{ A = a, B = b }[Hello, world!]{}`, or  
`\Ccool{ A = a, B = b } Hello, world!`

2.

Don't: `$\langle key_i \rangle <x$`.

Do: `$\langle key_i \rangle \{<\}x$`

3.

Don't: `[a, b)`

Do: `{[ ]a, b{ }}`

4.

Don't: `\Ccool{ F = \cal F }`.

Do: `\Ccool{ F = \cal{F} }` or `\Ccool{ F = \mathcal{F} }`

5. Also see [Part III, section 3](#)

## Part II

# Listing

### Listing 1.

```
% \CcoolVers
%
```

---

2020/04/12 v1.8 cool — A tool for encoding mathematical notation

### Listing 2. Preamble<sup>a</sup>

---

<sup>a</sup>These are the settings to replicate the listings. For exhaustivity, check the **documentation** section of `ccool.dtx`.

```
% \usepackage{amsmath, amsthm, commath}
% \usepackage[T1]{fontenc}% \char`[
%
```

### Listing 3. Separators

```
% \CcoolOption{
% ^^A% spaces betw. inner and outer brackets matter!->
% Separ={\ \char`@\ }{\ \% \ }{\ \char`@\ }}
% \Ccool<Test>{ X = x, Y = y }*[\]
% { X = x, Y = y, Z = z }*[\]
% { X = x, Y = y }*s{\ \&\ }{[\]}
% { X = x, Y = y }*s{\ \&\ }{,\ }{[\]}
% { X = x, Y = y, Z = z }*s{\ \&\ }{[\]}
% { X = x, Y = y, Z = z }*s{\ \&\ }{,\ }{[\]}
% { X = x, Y = y, Z = z }*s{\ \&\ }{,\ }{,\ }{[\]}
%
```

---

```
x @ y
x \% y @ z
x & y
x & y
x & y & z
x, y & z
x, y & z
```

### Listing 4. Hello, world!<sup>a</sup>

---

<sup>a</sup>If this looks arcane, it's for the purpose of testing.

```
% \CcoolOption{ Separ = {\}{.}{.}}, Outer = {####1} }
% \CcoolOption{ Write = \BooleanTrue }
```



```

% \Ccool<Test>
% { KeyA = {.}, KeyB = {!}, KeyC = {\%} }[]
% { KeyD = {d}, KeyE = {\%} }[]i{\#1\}
% { KeyF = {H}, KeyG = {e}, KeyH = {l} }*[]
% { KeyI = {\%}, KeyJ = {\%}, KeyK = {\%} }[.\{l\}.\{o\}]
% { KeyL = {l}, KeyM = {\char`[]}, KeyN = {\char`[]}] }[]
% { KeyO = {o}, KeyP = {\%}, KeyQ = {\%} }[{, \}]
% { KeyR = {w}, KeyS = {o}, KeyT = {r} }*s{{}{}}o{{\char`[]\#1}[]
% { KeyU = {\%}, KeyV = {\%}, KeyW = {\%} }[]
% { KeyX = {\%}, KeyY = {\%}, KeyZ = {\KeyB<Test>} }\nobreak
% \KeyL<Test>\KeyD<Test>\KeyZ<Test>\KeyN<Test>\
% \CcoolOption{ Write = \BooleanFalse }
%

```

{H}.\{e}.\{l}.\{l}.\{o}, [world!]

#### Listing 5. Listing 4 read from file.

```

% \CcoolRead
% \KeyF<Test>\KeyA<Test>\nobreak
% \KeyG<Test>\KeyA<Test>\nobreak
% \KeyH<Test>\KeyA<Test>\nobreak
% \KeyH<Test>\KeyA<Test>\nobreak
% {\{} \nobreak \KeyO<Test>{\}}, {\} \nobreak
% \KeyM<Test>\KeyR<Test>\nobreak
% \KeyO<Test>\nobreak
% \KeyT<Test>\nobreak
% \KeyL<Test>\nobreak
% \KeyD<Test>\nobreak
% \KeyZ<Test>\nobreak
% \KeyN<Test>\nobreak
%

```

{H}.\{e}.\{l}.\{l}.\{o}, [world!]

#### Listing 6. Probability space

```

% \CcoolOption{ Write = \BooleanTrue }
% \Ccool[Let~]
% { Space = \Omega, Field = \mathcal{F}, Meas = \mathcal{P} }
% *s{{,}}o{{\#1\}}
% [~denote the probability space, where~]{ PowerSet = { 2^{\Space} } }
% [{$\Field\subset \PowerSet$.}
% {
% \CcoolOption{ Write = \BooleanFalse }
%

```

Let  $\{\Omega, \mathcal{F}, \mathcal{P}\}$  denote the probability space, where  $\mathcal{F} \subset 2^\Omega$ .

Listing 7. Listing 6 read from file.

```
% \CcoolRead \tab $\Omega$ $\mathcal{F}$ $\mathcal{P}$
%
```

$$\Omega \mathcal{F} \mathcal{P}$$

Listing 8. Mittelwertsatz für  $n$  Variable[1, 17.3]

```
% \CcoolOption{ Write = \BooleanTrue }
% \newtheorem{theorem}{Theorem}
% \AfterEndEnvironment{theorem}{\CcoolHook}
% \Ccool i{\mathbb{#1}}
% { N = { N } , R = { R } }+[]
% { Grad = { \operatorname{grad} } }+
% [\begin{theorem}
% [Mittelwertsatz f\"ur $n$ Variable]Es~sei~]
% { OffMenge = {D}, Ci = {C^{1}}, Strecke = { [x_0,x] } }+
% [$n\in\mathbb{N}, \sim\mathcal{O}ffMenge\subseteq\mathbb{R}^n$ eine offene Menge und
% $f\in C^1(\mathcal{O}ffMenge, \mathbb{R})$.
% Dann gibt es auf jeder Strecke $\mathcal{S}\subseteq\mathcal{O}ffMenge$ einen
% Punkt $\xi\in\mathcal{S}$,~]
% { Steig = { \frac{ f(x)-f(x_0) }{ x-x_0 } }, Punkt = { \xi } }+
% [so dass gilt
% \begin{equation*}
% \Steig = \operatorname{grad} f(\Punkt)^{\top}
% \end{equation*}
% \end{theorem}]
% {}
% (Check: $N$, $\Punkt$)
% \CcoolOption{ Write = \BooleanFalse }
%
```

**Theorem 1 (Mittelwertsatz für  $n$  Variable)** Es sei  $n \in \mathbb{N}$ ,  $D \subseteq \mathbb{R}^n$  eine offene Menge und  $f \in C^1(D, \mathbb{R})$ . Dann gibt es auf jeder Strecke  $[x_0, x] \subset D$  einen Punkt  $\xi \in [x_0, x]$ , so dass gilt

$$\frac{f(x) - f(x_0)}{x - x_0} = \operatorname{grad} f(\xi)^{\top}$$

(Check:  $\mathbb{N}$ ,  $\xi$ )

Listing 9. Listing 8 read from file.

```
% \CcoolRead \tab $N$ $R$ $\mathcal{O}ffMenge$ $\mathcal{C}i$ $\mathcal{S}trecke$
%
```

$$\mathbb{N} \mathbb{R} D C^1 [x_0, x]$$

#### Listing 10. Lambda expression.

```
% \CcoolOption{ Write = \BooleanTrue }
% \Ccool{ EvalAt = \CcoolLambda{(#1)}, ApplyOp =
  \CcoolLambda[2]{#1[#2]} }
% [Supposons une fonction  $f$ \EvalAt{t}$, et \etudions le probl\eme
  o\`u la fonctionnelle  $S$ \ApplyOp{S}{f}$ est donn\`ee par\dots]{ }
% \CcoolOption{ Write = \BooleanFalse }
%
```

Supposons une fonction  $f(t)$ , et étudions le problème où la fonctionnelle  $S[f]$  est donnée par...

#### Listing 11. Listing 10 read from file.

```
% \CcoolRead \tab  $f$ \EvalAt{t}$,  $S$ \ApplyOp{S}{f}$
%
```

$$f(t), S[f]$$

#### Listing 12. CUSUM statistic

```
% \newtheorem{definition}{Definition}
% \AfterEndEnvironment{definition}{\CcoolHook}
%
% \CcoolOption{ Write = \BooleanTrue }
% \Ccool{ SuchThat = { ;~ }, Time = { t }, Process = { \xi }, StopT =
  { T }, EvalAt = \CcoolLambda{(#1)} }
% [The CUSUM statistic process and the corresponding one-sided CUSUM
  stopping time are defined as follows:
% \begin{definition}\label{the CUSUM statistic}. Let~]
% { Scale = { \lambda }, Real = {\mathcal{R}} }+*s{{~\in~}}[~and~]
% { CUSUMthresh = { \nu } }+*o{#1\in\Real^{+}}$.}
% [-Define the following processes:]
% { LogWald = { u }, CUSUMst = { \StopT_{c} }, CUSUM = { y },
  LogWaldInf = { m } }+
% [\begin{enumerate}
% \item{ $\LogWald_{\Time}\EvalAt{ \Scale } = \Scale\Process_{\Time}
  - \frac{1}{2}\Scale^2\Time$ ;
%  $\LogWaldInf_{\Time}\EvalAt{ \Scale } = \inf_{0 \leq s \leq \Time}
  \CUSUM_{\Time}\EvalAt{ \Scale }$ .}
% \item{ $\CUSUM_{\Time}\EvalAt{ \Scale } =
  \LogWaldInf_{\Time}\EvalAt{ \Scale } - \LogWald_{\Time}\EvalAt{
  \Scale } \geq 0$ , which is the CUSUM statistic process.}
% \item{ $\CUSUMst \EvalAt{ \Scale, \LogWaldInf } = \inf\left[ \Time
  \geq 0 \text{ \textit{SuchThat} } \CUSUM_{\Time}\EvalAt{\Scale} \geq \LogWaldInf
  \right]$ , which is the CUSUM stopping time.}
% \end{enumerate}\end{definition}\par{}
%
% (Check:  $\Scale$ ,  $\CUSUM$ )
% \CcoolOption{ Write = \BooleanFalse }
```

%

The CUSUM statistic process and the corresponding one-sided CUSUM stopping time are defined as follows:

**Definition 1** . Let  $\lambda \in \mathcal{R}$  and  $\nu \in \mathcal{R}^+$ . Define the following processes:

1.  $u_t(\lambda) = \lambda \xi_t - \frac{1}{2} \lambda^2 t$ ;  $m_t(\lambda) = \inf_{0 \leq s \leq t} y_s(\lambda)$ .
2.  $y_t(\lambda) = m_t(\lambda) - u_t(\lambda) \geq 0$ , which is the CUSUM statistic process.
3.  $T_c(\lambda, m) = \inf [t \geq 0; y_t(\lambda) \geq m]$ , which is the CUSUM stopping time.

(Check:  $\lambda, y$ )

Listing 13. Listing 12 read from file.

```
%      \CcoolRead \tab $Time$ $Process$ $Scale$ $Real$ $CUSUMthresh$
      $LogWald$ $CUSUMst$ $CUSUM$ $LogWaldInf$
%
```

$t \xi \lambda \mathcal{R} \nu u T_c y m$

## Part III

# Other

### 1 Acknowledgment

This work has benefited from Q&A's from the L<sup>A</sup>T<sub>E</sub>Xcommunity[6]. Specific attributions are made in the implementation and [References](#).

### 2 Install

Compiling `ccool.dtx`<sup>2</sup> will generate `ccool.sty` and `ccool.pdf`

### 3 Issue

1. **Don't:** `Inner={\####1\}`  
**Symptom:** `\CcoolRead` fails  
**Do:** `Inner={\char'\####1\char'}}`

### 4 Support

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

### 5 Testing

#### 5.1 Technicality

Not possible to compile-check the expansion of a certain class of macros against predefined values[8]. Instead, one can visually check [Part II](#), as generated in [section 2](#) on one's own machine, against that [of the repository](#) for the same version.

#### 5.2 Platform

1. 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

#### 5.3 Engine

1. pdfTeX 3.14159265-2.6-1.40.20 (TeX Live 2019)

#### 5.4 Results

1. `ccool v1.8` satisfactory on platform [1](#) and engine [1](#)

---

<sup>2</sup>Under Unix, `$tex ccool.dtx`

## 5.5 Other

Check [5] for using `ccool` with `lncs`

## References

- [1] Nick Setzer *The cool package*, 2005, <https://www.ctan.org/pkg/cool>
- [2] The L<sup>A</sup>T<sub>E</sub>X3 Project Team *The L<sup>A</sup>T<sub>E</sub>X3 interfaces*, 2019, <http://ftp.math.purdue.edu/mirrors/ctan.org/macros/latex/contrib/l3kernel/interface3.pdf>
- [3] Thomas F. Sturm *The tcolorbox package*, 2019, <http://www.texdoc.net/texmf-dist/doc/latex/tcolorbox/tcolorbox.pdf>
- [4] The L<sup>A</sup>T<sub>E</sub>X3 Project Team *The xparse package*, 2020, <http://ftp.math.purdue.edu/mirrors/ctan.org/macros/latex/contrib/l3packages/xparse.pdf>
- [5] Erwann Rogard and Olympia Hadjiliadis *Typesetting a math thesis with ccool*, 2020, <https://github.com/rogard/ccool/blob/master/thesis.pdf>
- [6] User `erwann` at [tex.stackexchange.com](https://tex.stackexchange.com), <https://tex.stackexchange.com/users/112708/erwann?tab=questions>
- [7] `@sean-allred`’s answer to “How to create lambda expressions?”, <https://tex.stackexchange.com/a/188053/112708>
- [8] “Checking a function’s expansion against a string”, <https://tex.stackexchange.com/a/534100>
- [9] `@frougon`’s answer to “Journaling calls to a function []”, <https://tex.stackexchange.com/a/536620>

## Part IV

# Implementation

```

1 <@@=ccool>
2 \NeedsTeXFormat{LaTeX2e}[2019/10/01]
3 \ExplSyntaxOn

```

### 1 aux

```

\__ccool_aux_inner_set:n #1: <code>

4 \cs_new_protected:Nn \__ccool_aux_inner_set:n
5 {
6   \cs_gset:Npn \__ccool_aux_inner:n ##1 {#1}
7   \cs_generate_variant:Nn \__ccool_aux_inner:n { e }
8 }

(End definition for \__ccool_aux_inner_set:n.)

\__ccool_aux_key:w #1: <key>
#2: <value>

9 \cs_new_protected:Npn \__ccool_aux_key:w #1 = #2 \q_stop
10 {
11   \seq_gput_right:Nx \g__ccool_aux_key_seq { \tl_trim_spaces:n{#1} }
12 }

(End definition for \__ccool_aux_key:w.)

\__ccool_aux_key:n #1: <key = value>

13 \cs_new_protected:Nn \__ccool_aux_key:n
14 {
15   \__ccool_aux_key:w #1 \q_stop
16 }

(End definition for \__ccool_aux_key:n.)

\__ccool_aux_key:N #1: <seq>

17 \cs_new_protected:Nn \__ccool_aux_key:N
18 {
19   \seq_gclear_new:N \g__ccool_aux_key_seq
20   \seq_map_function:NN #1 \__ccool_aux_key:n
21 }

(End definition for \__ccool_aux_key:N.)

\__ccool_aux_outer_set:n #1: <inline code>

22 \cs_new_protected:Nn \__ccool_aux_outer_set:n
23 {
24   \cs_gset:Npn \__ccool_aux_outer:n ##1 {#1}
25 }

(End definition for \__ccool_aux_outer_set:n.)

```

```

\__ccool_aux_prop:nn
26 \prop_new:N \g__ccool_aux_prop
27 \cs_new_protected:Nn \__ccool_aux_prop:nn
28 {
29   \prop_gput:Nnn \g__ccool_aux_prop{#1}{#2}
30 }
31 \cs_generate_variant:Nn \__ccool_aux_prop:nn { eo, ee, ex, xo, xe, xx }

(End definition for \__ccool_aux_prop:nn.)

```

```

\__ccool_aux_prop:w #1 : < key >
#2 : < value >

32 \tl_new:N \g__ccool_option_expans_tl
33 \cs_new_protected:Npn \__ccool_aux_prop:w #1 = #2 \q_stop
34 {
35   \exp_args:Nx
36   \use:c{\__ccool_aux_prop:\g__ccool_option_expans_tl}
37   { \tl_trim_spaces:n{#1} }
38   { \__ccool_aux_inner:n{ \tl_trim_spaces:n{#2} } }
39 }

(End definition for \__ccool_aux_prop:w.)

```

```

\__ccool_aux_prop:n #1 : < key = value >

40 \cs_new_protected:Nn \__ccool_aux_prop:n
41 {
42   \__ccool_aux_prop:w #1 \q_stop
43 }

(End definition for \__ccool_aux_prop:n.)

```

```

\__ccool_aux_prop:N #1 : < keyval list >

44 \cs_new_protected:Nn \__ccool_aux_prop:N
45 {
46   \prop_gclear_new:N \g__ccool_aux_prop
47   \seq_if_empty:NTF #1
48   { \c_empty_tl }
49   {
50     \seq_map_function:NN #1 \__ccool_aux_prop:n
51   }
52 }

(End definition for \__ccool_aux_prop:N.)

```

```

\__ccool_aux_separ:nn #1 : < int >
#2 : < tokens >

53 \cs_new:Nn \__ccool_aux_separ:nn
54 {
55   \int_case:nnTF {#1}
56   {
57     {1}
58     { \prg_replicate:nn{ 3 }{#2} }
59     {2}
60     {

```



```

61     { \use_i:nn #2 }
62     { \use_ii:nn #2 }
63     { \use_i:nn #2 }
64   }
65   {3}{#2}
66 }
67 { \c_empty_tl }
68 {
69   \msg_error:nnnn { __erw }
70   { separ }
71   { \exp_not:N \__ccool_aux_separ:nn }
72   {#2}
73 }
74 }
75 \cs_generate_variant:Nn \__ccool_aux_separ:nn { e }

```

(End definition for \\_\_ccool\_aux\_separ:nn.)

```

\__ccool_aux_separ:n #1: < tokens >
76 \cs_new:Nn \__ccool_aux_separ:n
77 {
78   \__ccool_aux_separ:en{ \tl_count:n{#1} }{#1}
79 }

```

(End definition for \\_\_ccool\_aux\_separ:n.)

```

\__ccool_aux_val:Nn #1: < seq >
#2: < tl var name >
80 \cs_new_protected:Nn \__ccool_aux_val:Nn
81 {
82   \seq_gclear_new:N \g__ccool_aux_val_seq
83   \__ccool_seq_from_prop:NNn \g__ccool_aux_val_seq #1 { \__ccool_prop_name:n{#2} }
84 }

```

(End definition for \\_\_ccool\_aux\_val:Nn.)

## 2 lambda

```

\__ccool_lambda:nn [7]
85 \cs_new_protected:Npn \__ccool_lambda:nn #1 #2
86 {
87   \exp_args:NNx
88   \DeclareDocumentCommand \__ccool_lambda_expression
89   { \prg_replicate:nn { #1 } { m } }
90   {#2}
91   \__ccool_lambda_expression
92 }

```

(End definition for \\_\_ccool\_lambda:nn.)

### 3 log

\\_ccool\_log\_close:

```

93 \iow_new:N \g__ccool_log_iow
94 \AtEndDocument{\iow_close:N \g__ccool_log_iow}
95 \bool_set_false:N \g__ccool_log_open_bool
96 \cs_new_protected:Nn \_ccool_log_close:
97 {
98   \iow_close:N \g__ccool_log_iow
99   \bool_gset_false:N \g__ccool_log_open_bool
100 }

```

(End definition for \\_ccool\_log\_close:.)

\\_ccool\_log\_open:

```

101 \tl_new:N \g__ccool_log_file_tl
102 \cs_new_protected:Nn \_ccool_log_open:
103 {
104   \tl_gset:Nx \g__ccool_log_to_tl{\g__ccool_log_file_tl}
105   \iow_open:Nn \g__ccool_log_iow {\g__ccool_log_to_tl}
106   \bool_gset_true:N \g__ccool_log_open_bool
107 }

```

(End definition for \\_ccool\_log\_open:.)

\\_ccool\_log\_read:n #1 :  $\langle path \rangle$

```

108 \cs_new_protected:Nn \_ccool_log_read:n
109 {
110   \file_input:n{#1}
111   \tl_log:n{read~from~#1}
112 }
113 \cs_generate_variant:Nn \_ccool_log_read:n { e }

```

(End definition for \\_ccool\_log\_read:n.)

\\_ccool\_log\_read:

```

114 \cs_new_protected:Nn \_ccool_log_read:
115 {
116   \_ccool_log_read:e{\g__ccool_log_to_tl}
117 }

```

(End definition for \\_ccool\_log\_read:.)

\\_ccool\_log\_write:n

```

118 \tl_new:N \g__ccool_log_to_tl
119 \cs_new_protected:Nn \_ccool_log_write:n
120 {
121   \bool_if:nTF{ \g__ccool_log_open_bool }
122   {
123     \iow_now:Nn \g__ccool_log_iow {#1}
124     \tl_log:n{ write~to~#1 }
125   }
126   { \msg_error:nnnn{ __ccool }{ iow }{ \g__ccool_log_iow } }
127 }
128 \cs_generate_variant:Nn \_ccool_log_write:n { e }

```

(End definition for \\_ccool\_log\_write:n.)

## 4 make\_key

```

__ccool_make_key:Nn #1 : < token >
#2 : < key >

129 \cs_new_protected:Nn \__ccool_make_key:Nn
130 {
131   \exp_args:NNx
132   \ProvideDocumentCommand{#1}
133   { D<>{\g__ccool_option_name_tl} }
134   {
135     \__ccool_prop_item:nn{#1}{#2}
136   }
137 }
138 \cs_generate_variant:Nn \__ccool_make_key:Nn {c}

(End definition for \__ccool_make_key:Nn.)

__ccool_make_key:n #1 : < key >

139 \cs_new_protected:Nn \__ccool_make_key:n
140 {
141   \__ccool_make_key:cn{#1}{#1}
142 }
143 \cs_generate_variant:Nn \__ccool_make_key:n { e }

(End definition for \__ccool_make_key:n.)

__ccool_make_key:N #1 : < seq >

144 \cs_new_protected:Nn \__ccool_make_key:N
145 {
146   \seq_map_function:NN #1 \__ccool_make_key:e
147 }

(End definition for \__ccool_make_key:N.)

```

## 5 make\_ccool

```

__ccool_make_ccool_exp:nnn

148 \cs_new_protected:Nn \__ccool_make_ccool_exp:nnn
149 {
150   \__ccool_aux_val:Nn \g__ccool_aux_key_seq {#1}
151   \__ccool_aux_outer_set:n{#3}
152   \__ccool_aux_outer:n
153   {
154     \exp_args:NNf
155     \__ccool_seq_use:Nn
156     \g__ccool_aux_val_seq
157     {#2}
158   }
159 }

(End definition for \__ccool_make_ccool_exp:nnn.)

```

\\_ccool\\_make\\_ccool\\_key:nnn

```

160 \cs_new_protected:Nn \_ccool\_make\_ccool\_key:nnn
161 {
162   \_ccool\_prop\_if\_exist:nTF{#1}
163   { \c\_empty\_tl }
164   { \_ccool\_prop\_new:n{#1} }
165   \exp\_args:No \_ccool\_aux\_inner\_set:n{#2}
166   \seq\_set\_from\_clist:Nn \g\_ccool\_aux\_keyval\_seq {#3}
167   \_ccool\_aux\_prop:N \g\_ccool\_aux\_keyval\_seq
168   \_ccool\_prop\_append:Nn \g\_ccool\_aux\_prop {#1}
169   \_ccool\_aux\_key:N \g\_ccool\_aux\_keyval\_seq
170   \_ccool\_make\_key:N \g\_ccool\_aux\_key\_seq
171 }

```

(End definition for \\_ccool\\_make\\_ccool\\_key:nnn.)

\\_ccool\\_make\\_ccool\\_sideeffect:nnn [9]

```

172 \cs_new_protected:Nn \_ccool\_make\_ccool\_sideeffect:nnn
173 {
174   \_ccool\_make\_ccool\_key:nnn{#1}{#2}{#3}
175   \bool\_if:nTF{ \g\_ccool\_log\_open\_bool }
176   {%~A https://tex.stackexchange.com/questions/536597
177     \_ccool\_log\_write:n
178     {
179       \begin{group}
180       \def \_ccool\_log\_entry { \Ccool<#1>i{#2}{#3} } \expandafter
181       \end{group} \_ccool\_log\_entry
182     }
183   }{\c\_empty\_tl}
184 }

```

(End definition for \\_ccool\\_make\\_ccool\\_sideeffect:nnn.)

\\_ccool\\_make\\_ccool:nnnn #1 : < token list >  
 #2 : < seq<sub>1</sub> >  
 #3 : < seq<sub>2</sub> >  
 #4 : < prop >

```

185 \def\CcoolHook{\c\_empty\_tl}
186 \cs_new_protected:Npn \_ccool\_make\_ccool:nnnn #1 #2 #3 #4
187 {
188   \exp\_args:NNx \DeclareDocumentCommand \Ccool
189   {%~A      2      3      4 5 6      7 8      9
190     D<>{#1} +o E{ i }{{#2}} m t+ s E{ s o }{{#3}{#4}} +o
191   }
192   {
193     \IfValueT{##2}{##2}
194     \_ccool\_make\_ccool\_sideeffect:nnn{##1}{##3}{##4}
195     \IfBooleanT{##6}
196     {
197       \_ccool\_make\_ccool\_exp:nnn{##1}{##7}{##8}
198     }
199     \bool\_if:nTF{##5}
200     {
201       \gappto{\CcoolHook}

```

```

202     {
203       \__ccool_make_ccool_sideeffect:nnn{##1}{##3}{##4}
204     }
205   }
206   {\c_empty_tl}
207   \IfValueT{##9}
208   {
209     \exp_not:n{ \Ccool<##1>[##9] }
210   }
211 }
212 }

```

(End definition for \\_\_ccool\_make\_ccool:nnnn.)

## 6 msg

```

213 \msg_new:nnn {\__ccool}{ generic }{#1}
214 \msg_new:nnn {\__ccool}{ iow }{#1~is~closed~can't~write}
215 \msg_new:nnn {\__ccool}{ keyonly }{#1~does~not~take~values;~keyval~is~#2}
216 \msg_new:nnn {\__ccool}{ keywrong }{#1~does~not~recognize~key~#2}
217 \msg_new:nnn {\__ccool}{ separ }{#1~expects~1~to~3~items,~#2}
218 \msg_new:nnn {\__ccool}{ unset }{#1~unset}

```

## 7 option

\\_\_ccool\_aux\_inner:n #1 : *<code>*

```

219 \cs_new_protected:Nn \__ccool_option_inner:n
220 {
221   \tl_gset:Nn \g__ccool_option_inner_tl {#1}
222 }
223 \__ccool_option_inner:n
224 {
225   \msg_warning:nnn{ __ccool }{ unset }{ \exp_not:N \g__ccool_option_inner_tl }
226 }

```

(End definition for \\_\_ccool\_aux\_inner:n.)

\\_\_ccool\_option\_name:n #1 : *<token list>*

```

227 \cs_new:Nn \__ccool_option_name:n
228 {
229   \tl_gset:Nn \g__ccool_option_name_tl{#1}
230 }
231 \__ccool_option_name:n
232 {
233   \msg_error:nnx{ __ccool }
234   { generic }
235   { \exp_not:N\g__ccool_option_name_tl~undefined }
236 }

```

(End definition for \\_\_ccool\_option\_name:n.)

```

\__ccool_option_outer:n #1 :  $\langle inline code \rangle$ 
237 \cs_new_protected:Nn \__ccool_option_outer:n
238 {
239   \tl_gset:Nn \g__ccool_option_outer_tl {#1}
240 }
241 \__ccool_option_outer:n
242 {
243   \msg_warning:nnn{ __ccool }{ unset }{ \exp_not:N \g__ccool_option_outer_tl }
244 }

```

(End definition for \\_\_ccool\_option\_outer:n.)

```

\__ccool_option_separ:n #1 : { $\langle tl_1 \rangle$ }{ $\langle tl_2 \rangle$ }{ $\langle tl_3 \rangle$ }
245 \cs_new_protected:Nn \__ccool_option_separ:n
246 {
247   \cs_gset:Npn \g__ccool_option_separ_tl {#1}
248 }
249 \__ccool_option_separ:n
250 {
251   \msg_warning:nnn{ __ccool }{ unset }{ \exp_not:N \g__ccool_option_separ_tl }
252 }

```

(End definition for \\_\_ccool\_option\_separ:n.)

## 8 prop

```

\__ccool_prop_append:NN #1 :  $\langle prop_1 \rangle$ 
#2 :  $\langle prop_2 \rangle$ 
253 \cs_new_protected:Npn \__ccool_prop_append:NN #1 #2
254 {
255   \cs_set:Nn \__ccool_prop_append:nn
256   {
257     \prop_gput:Nnx #1 {##1}{ \prop_item:Nn #2{##1} }
258   }
259   \prop_map_function:NN #2 \__ccool_prop_append:nn
260 }
261 \cs_generate_variant:Nn \__ccool_prop_append:NN { cN }

```

(End definition for \\_\_ccool\_prop\_append:NN.)

```

\__ccool_prop_append:Nn #1 :  $\langle prop \rangle$ 
#2 :  $\langle tl var name \rangle$ 
262 \cs_new_protected:Nn \__ccool_prop_append:Nn
263 {
264   \__ccool_prop_append:cN{ \__ccool_prop_name:n {#2} } #1
265 }

```

(End definition for \\_\_ccool\_prop\_append:Nn.)

```

\__ccool_prop_clear_new:n #1 :  $\langle tl var name \rangle$ 
266 \cs_new_protected:Nn \__ccool_prop_clear_new:n
267 {
268   \exp_args:No \prop_clear_new:c{ \__ccool_prop_name:n {#1} }
269 }

```

(End definition for \\_ccool\_prop\_clear\_new:n.)

```
\_ccool_prop_clear_new_map:n #1 : < keyval list >
270 \cs_new_protected:Nn \_ccool_prop_clear_new_map:n
271 {
272   \seq_set_from_clist:Nn \g__ccool_aux_key_seq {#1}
273   \seq_map_function:NN \g__ccool_aux_key_seq \_ccool_prop_clear_new:n
274 }
```

(End definition for \\_ccool\_prop\_clear\_new\_map:n.)

```
\_ccool_prop_if_exist:nTF #1 : < tl_1 >
#2 : < tl_2 >
#3 : < tl_3 >
275 \cs_new:Nn \_ccool_prop_if_exist:nTF
276 {
277   \prop_if_exist:cTF{ \_ccool_prop_name:n {#1} }{#2}{#3}
278 }
```

(End definition for \\_ccool\_prop\_if\_exist:nTF.)

```
\_ccool_prop_item:nn #1 : < tl var name >
#2 : < key >
279 \cs_new:Nn \_ccool_prop_item:nn
280 {
281   \prop_item:cn { \_ccool_prop_name:n {#1} } {#2}
282 }
```

(End definition for \\_ccool\_prop\_item:nn.)

```
\_ccool_prop_name:n #1 : < tl var name >
283 \cs_new:Npn \_ccool_prop_name:n #1{ __ccool_#1 }
```

(End definition for \\_ccool\_prop\_name:n.)

```
\_ccool_prop_new:n #1 : < tl var name >
284 \cs_new_protected:Nn \_ccool_prop_new:n
285 {
286   \prop_new:c{ \_ccool_prop_name:n {#1} }
287 }
```

(End definition for \\_ccool\_prop\_new:n.)

## 9 seq

```

\__ccool_seq_from_prop:NNn #1: < seq1 >
#2: < seq2 > (keys)
#3: < prop >

288 \cs_new_protected:Nn \__ccool_seq_from_prop:NNn
289 {
290   \cs_set_protected:Nn \__ccool_seq_from_prop:n
291   {
292     \seq_gput_right:No #1 { \prop_item:cn{#3}{##1} }
293   }
294   \seq_map_function:NN #2 \__ccool_seq_from_prop:n
295 }

```

(End definition for \\_\_ccool\_seq\_from\_prop:NNn.)

```

\__ccool_erw_seq_use:Nn

296 % \begin{arguments}
297 % \item \meta{ seq }
298 % \item \meta{ tokens }
299 % \end{arguments}
300 \cs_new:Nn \__ccool_seq_use:Nn
301 {
302   \exp_last_unbraced:NNf
303   \seq_use:Nnnn #1
304   \__ccool_aux_separ:n{#2}
305 }

```

(End definition for \\_\_ccool\_erw\_seq\_use:Nn.)

## 10 Front-end

```

306 \keys_define:nn { __ccool }
307 {
308   Expans .multichoices:nn =
309   { eo, ee, ex, xo, xe, xx }
310   { \tl_gset_eq:NN \g__ccool_option_expans_tl \l_keys_choice_tl },
311   Expans .default:n = { xo },
312   Expans .initial:n = { xo },
313   File .code:n = { \tl_gset:Nn \g__ccool_log_file_tl{ \exp_not:n{ #1 } } },
314   File .default:n = { ccool\pdfcreationdate },
315   File .initial:n = { ccool\pdfcreationdate },
316   Name .code:n={
317     \__ccool_option_name:n{#1}
318     \exp_last_unbraced:Nf
319     \__ccool_make_ccool:nnnn
320     {
321       { \g__ccool_option_name_tl }
322       { \g__ccool_option_inner_tl }
323       { \g__ccool_option_separ_tl }
324       { \g__ccool_option_outer_tl }
325     }
326   },

```



```

327 Name .value_required:n = false,
328 Name .default:n = { Math },
329 Name .initial:n = { Math },
330 Inner .code:n={
331   \__ccool_option_inner:n{#1}
332   \exp_last_unbraced:Nf
333   \__ccool_make_ccool:nnnn
334   {
335     { \g__ccool_option_name_tl }
336     { \g__ccool_option_inner_tl }
337     { \g__ccool_option_separ_tl }
338     { \g__ccool_option_outer_tl }
339   }
340 },
341 Inner .value_required:n = false,
342 Inner .default:n = {####1},
343 Inner .initial:n = {####1},
344 Outer .code:n={
345   \__ccool_option_outer:n{#1}
346   \exp_last_unbraced:Nf
347   \__ccool_make_ccool:nnnn
348   {
349     { \g__ccool_option_name_tl }
350     { \g__ccool_option_inner_tl }
351     { \g__ccool_option_separ_tl }
352     { \g__ccool_option_outer_tl }
353   }
354 },
355 Outer .value_required:n = false,
356 Outer .default:n = { \ensuremath{####1} },
357 Outer .initial:n = { \ensuremath{####1} },
358 Write .code:n = {
359   \bool_if:nTF{#1}
360   {\__ccool_log_open:}
361   {\__ccool_log_close:}
362 },
363 Write .value_required:n = false,
364 Write .default:n = \BooleanFalse,
365 Write .initial:n = \BooleanFalse,
366 Separ .code:n={
367   \__ccool_option_separ:n{#1}
368   \exp_last_unbraced:Nf
369   \__ccool_make_ccool:nnnn
370   {
371     { \g__ccool_option_name_tl }
372     { \g__ccool_option_inner_tl }
373     { \g__ccool_option_separ_tl }
374     { \g__ccool_option_outer_tl }
375   }
376 },
377 Separ .value_required:n = false,
378 Separ .default:n = { {\ }and{\ } } { ,{\ } } { ,{\ }and{\ } },
379 Separ .initial:n = { {\ }and{\ } } { ,{\ } } { ,{\ }and{\ } }
380 }

```

**\CcoolClear** #1 :  $\langle \text{tl var name} \rangle$

```

381 \NewDocumentCommand{ \CcoolClear }
382 { D<>{\g__ccool_option_name_tl} }
383 {
384   \__ccool_prop_clear_new_map:n{#1}
385 }

```

(End definition for \CcoolClear. This function is documented on page 5.)

**\CcoolLambda**

```

386 \ProvideDocumentCommand \CcoolLambda { 0{1} m }
387 {
388   \__ccool_lambda:nn { #1 } { #2 }
389 }

```

(End definition for \CcoolLambda. This function is documented on page 5.)

**\CcoolOption**

```

390 \NewDocumentCommand{ \CcoolOption }
391 { m }
392 {
393   \keys_set:nn{ __ccool }{#1}
394 }

```

(End definition for \CcoolOption. This function is documented on page 5.)

**\CcoolRead**

```

395 \NewDocumentCommand{\CcoolRead}
396 {o}
397 {
398   \IfValueTF{#1}
399   {\__ccool_log_read:e{#1}}
400   {\__ccool_log_read:}
401 }

```

(End definition for \CcoolRead. This function is documented on page 6.)

**\CcoolVers**

```

402 \NewDocumentCommand{\CcoolVers}
403 {}
404 {\use:c{ver@ccool.sty}}

```

(End definition for \CcoolVers. This function is documented on page 6.)

## 11 Misc

```

405 \ExplSyntaxOff

```

# Change History

v1.0	General: Initial version . . . . .	14	Added: <b>Expans</b> (for debugging' sake, but...) . . . . .	14
v1.1	General: Added: <b>Save</b> . . . . .	14	Added: Listing 1., 2., and 3. . . . .	14
	Added: Listing 1., 2., 3., 4., 6., and 9. . . . .	14	Deleted: Listing 1., and 2. . . . .	14
	Added: <b>\OpsRestore</b> . . . . .	14	Replaced: $s\{\langle tl_3 \rangle\}\{\langle tl_4 \rangle\}\{\langle tl_5 \rangle\}$ by $s\{\{\langle tl_3 \rangle\} \{\langle tl_3 \rangle\}\{\langle tl_4 \rangle\} \{\langle tl_3 \rangle\}\{\langle tl_4 \rangle\}\{\langle tl_5 \rangle\}\}$ . . . . .	14
	Added: <b>\OpsTest</b> . . . . .	14		
	Deleted: Listing 1-5 from v1.0 . . .	14	v1.5	General: Added: <b>File</b> . . . . .
	Fixed: apparent anomaly in v1.0's Listing 4, see Listing 3 . . . . .	14		Deleted: dependence on <b>datetime</b> . .
	Replaced: <b>\OpsOptions</b> by <b>\OpsOption</b> . .	14	v1.6	General: Added: Listing 2 (preamble) .
	Replaced: $\{\langle kvl_2 \rangle\}$ by $\langle kvl\_2 \rangle$ given that option type G not recommended[4] .	14		Renamed: <b>\OpsClear</b> to <b>\CcoolClear</b> . . . . .
	Replaced: <b>GenericObject</b> by <b>Name</b> . . .	14		Renamed: <b>\OpsDebug</b> to <b>\CcoolDebug</b> . . . . .
	Replaced: <b>Separators</b> by <b>Separ</b> . . .	14		Renamed: <b>\OpsHook</b> to <b>\CcoolHook</b> . . . . .
	Revamped: much of the implementation . . . . .	14		Renamed: <b>\OpsOption</b> to <b>\CcoolOption</b> . . . . .
v1.2	General: Deleted: <b>\OpsTest</b> . . . . .	14		Renamed: <b>\OpsRead</b> to <b>\CcoolRead</b> . . . . .
	Deleted: $\langle kvl_2 \rangle$ and $\langle code_2 \rangle$ . . . . .	14		Renamed: <b>\Ops</b> to <b>\Ccool</b> . . . . .
	Deleted: Listing 2-3 from v1.1. . . .	14		Renamed: <b>oops</b> to <b>ccool</b> (better describes the purpose) . . . . .
	Replaced: <b>\OpsClear</b> $\{\langle tl_1 \rangle\}$ by <b>\OpsClear</b> $[\langle keyval list \rangle]$ . . . . .	14	v1.7	General: Added: Legends to listings . .
	Replaced: <b>\Restore</b> by <b>\Read</b> . . . .	14		Added: Listing 12 (CUSUM) . . . . .
	Replaced: <b>\Save</b> by <b>\Write</b> . . . . .	14		Deleted: <b>\CcoolDebug</b> . . . . .
v1.3	General: Replaced: <b>\OpsNew</b> by <b>\Ops</b> . .	14		Deleted: Listing 5 from v1.6 . . . . .
	Replaced: $\{\langle tl_1 \rangle\}$ and $[\langle tl_1 \rangle]$ by $\langle tl_1 \rangle$ . . . . .	14	v1.8	General: Added: <b>\CcoolLambda</b> . . . .
v1.4	General: Added: <b>section 9</b> . . . . .	14		Added: <b>\CcoolVers</b> . . . . .
	Added: <b>\OpsDebug</b> . . . . .	14		Added: Listing 10, Listing 11 . . . .
	Added: <b>\OpsHook</b> . . . . .	14		Added: Listing 1 . . . . .

# 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>Symbols</b>	$\langle kvl_1 \rangle$ (option) . . . . .	4
* (option) . . . . .	$\langle tl_1 \rangle$ (option) . . . . .	4
+ (option) . . . . .	$\langle tl_2 \rangle$ (option) . . . . .	4
$\backslash \langle key_i \rangle$ . . . . .	$\langle tl_3 \rangle$ (option) . . . . .	5
$\langle code_1 \rangle$ (option) . . . . .	$\langle tl_4 \rangle$ (option) . . . . .	5
$\langle code_2 \rangle$ (option) . . . . .	$\langle tl_5 \rangle$ (option) . . . . .	5

$\langle \text{tl}_6 \rangle$ (option) . . . . .	5	<code>\__ccool_log_open:</code> . . . . .	101, 360
Expans (option) . . . . .	5	<code>\g__ccool_log_open_bool</code> . . . . .	95, 99, 106, 121, 175
File (option) . . . . .	6	<code>\__ccool_log_read:</code> . . . . .	114, 400
Inner (option) . . . . .	6	<code>\__ccool_log_read:n</code> . . . . .	108, 116, 399
Name (option) . . . . .	6	<code>\g__ccool_log_to_tl</code> . . . . .	104, 105, 116, 118
Outer (option) . . . . .	6	<code>\__ccool_log_write:n</code> . . . . .	118, 177
Separ (option) . . . . .	6	<code>\__ccool_make_ccool:nnnn</code> . . . . .	185, 319, 333, 347, 369
Write (option) . . . . .	6	<code>\__ccool_make_ccool_exp:nnn</code> . . . . .	148, 197
<code>\_</code> . . . . .	378, 379	<code>\__ccool_make_ccool_key:nnn</code> . . . . .	160, 174
<b>A</b>			
<code>\AtEndDocument</code> . . . . .	94	<code>\__ccool_make_ccool_sideeffect:nnn</code> . . . . .	172, 194, 203
<b>B</b>			
<code>\begin</code> . . . . .	296	<code>\__ccool_make_key:N</code> . . . . .	144, 170
<code>\begingroup</code> . . . . .	179	<code>\__ccool_make_key:n</code> . . . . .	139, 146
bool commands:		<code>\__ccool_make_key:Nn</code> . . . . .	129, 141
<code>\bool_gset_false:N</code> . . . . .	99	<code>\g__ccool_option_expans_tl</code> . . . . .	32, 36, 310
<code>\bool_gset_true:N</code> . . . . .	106	<code>\__ccool_option_inner:n</code> . . . . .	219, 223, 331
<code>\bool_if:nTF</code> . . . . .	121, 175, 199, 359	<code>\g__ccool_option_inner_tl</code> . . . . .	221, 225, 322, 336, 350, 372
<code>\bool_set_false:N</code> . . . . .	95	<code>\__ccool_option_name:n</code> . . . . .	227, 317
<code>\BooleanFalse</code> . . . . .	364, 365	<code>\g__ccool_option_name_tl</code> . . . . .	133, 229, 235, 321, 335, 349, 371, 382
<b>C</b>			
<code>\Ccool</code> . . . . .	1, 4, 4, 5, 5, 180, 188, 209	<code>\__ccool_option_outer:n</code> . . . . .	237, 345
ccool internal commands:		<code>\g__ccool_option_outer_tl</code> . . . . .	239, 243, 324, 338, 352, 374
<code>\__ccool_aux_inner:n</code> . . . . .	6, 7, 38, 219	<code>\__ccool_option_separ:n</code> . . . . .	245, 367
<code>\__ccool_aux_inner_set:n</code> . . . . .	4, 165	<code>\g__ccool_option_separ_tl</code> . . . . .	247, 251, 323, 337, 351, 373
<code>\__ccool_aux_key:N</code> . . . . .	17, 169	<code>\__ccool_prop_append:NN</code> . . . . .	253, 264
<code>\__ccool_aux_key:n</code> . . . . .	13, 20	<code>\__ccool_prop_append:Nn</code> . . . . .	168, 262
<code>\__ccool_aux_key:w</code> . . . . .	9, 15	<code>\__ccool_prop_append:nn</code> . . . . .	255, 259
<code>\g__ccool_aux_key_seq</code> . . . . .	11, 19, 150, 170, 272, 273	<code>\__ccool_prop_clear_new:n</code> . . . . .	266, 273
<code>\g__ccool_aux_keyval_seq</code> . . . . .	166, 167, 169	<code>\__ccool_prop_clear_new_map:n</code> . . . . .	270, 384
<code>\__ccool_aux_outer:n</code> . . . . .	24, 152	<code>\__ccool_prop_if_exist:nTF</code> . . . . .	162, 275
<code>\__ccool_aux_outer_set:n</code> . . . . .	22, 151	<code>\__ccool_prop_item:nn</code> . . . . .	135, 279
<code>\g__ccool_aux_prop</code> . . . . .	26, 29, 46, 168	<code>\__ccool_prop_name:n</code> . . . . .	83, 264, 268, 277, 281, 283, 286
<code>\__ccool_aux_prop:N</code> . . . . .	44, 167	<code>\__ccool_prop_new:n</code> . . . . .	164, 284
<code>\__ccool_aux_prop:n</code> . . . . .	40, 50	<code>\__ccool_seq_from_prop:n</code> . . . . .	290, 294
<code>\__ccool_aux_prop:nn</code> . . . . .	26	<code>\__ccool_seq_from_prop:NNn</code> . . . . .	83, 288
<code>\__ccool_aux_prop:w</code> . . . . .	32, 42	<code>\__ccool_seq_use:Nn</code> . . . . .	155, 300
<code>\__ccool_aux_separ:n</code> . . . . .	76, 304	<code>\CcoolClear</code> . . . . .	1, 5, 381
<code>\__ccool_aux_separ:nn</code> . . . . .	53, 78	<code>\CcoolHook</code> . . . . .	2, 4, 5, 185, 201
<code>\__ccool_aux_val:Nn</code> . . . . .	80, 150	<code>\CcoolLambda</code> . . . . .	2, 5, 386
<code>\g__ccool_aux_val_seq</code> . . . . .	82, 83, 156	<code>\CcoolOption</code> . . . . .	2, 5, 390
<code>\__ccool_erw_seq_use:Nn</code> . . . . .	296	<code>\CcoolRead</code> . . . . .	2, 6, 13, 395
<code>\__ccool_lambda:nn</code> . . . . .	85, 388	<code>\CcoolVers</code> . . . . .	2, 6, 402
<code>\__ccool_lambda_expression</code> . . . . .	88, 91	cs commands:	
<code>\__ccool_log_close:</code> . . . . .	93, 361	<code>\cs_generate_variant:Nn</code> . . . . .	7, 31, 75, 113, 128, 138, 143, 261
<code>\__ccool_log_entry</code> . . . . .	180, 181	<code>\cs_gset:Npn</code> . . . . .	6, 24, 247
<code>\g__ccool_log_file_tl</code> . . . . .	101, 104, 313	<code>\cs_new:Nn</code> . . . . .	53, 76, 227, 275, 279, 300
<code>\g__ccool_log_iow</code> . . . . .	93, 94, 98, 105, 123, 126		

<code>\cs_new:Npn</code> . . . . .	283	<code>\keys_define:nn</code> . . . . .	306
<code>\cs_new_protected:Nn</code> . . . . .	4, 13, 17, 22, 27, 40, 44, 80, 96, 102, 108, 114, 119, 129, 139, 144, 148, 160, 172, 219, 237, 245, 262, 266, 270, 284, 288	<code>\keys_set:nn</code> . . . . .	393
<code>\cs_new_protected:Npn</code> . . . . .	9, 33, 85, 186, 253	<b>M</b>	
<code>\cs_set:Nn</code> . . . . .	255	<code>\meta</code> . . . . .	297, 298
<code>\cs_set_protected:Nn</code> . . . . .	290	msg commands:	
<b>D</b>		<code>\msg_error:nnn</code> . . . . .	126, 233
<code>\DeclareDocumentCommand</code> . . . . .	88, 188	<code>\msg_error:nnnn</code> . . . . .	69
<code>\def</code> . . . . .	180, 185	<code>\msg_new:nnn</code> 213, 214, 215, 216, 217, 218	
<b>E</b>		<code>\msg_warning:nnn</code> . . . . .	225, 243, 251
<code>\end</code> . . . . .	299	<b>N</b>	
<code>\endgroup</code> . . . . .	181	<code>\NeedsTeXFormat</code> . . . . .	2
<code>\ensuremath</code> . . . . .	356, 357	<code>\NewDocumentCommand</code> . . . . .	381, 390, 395, 402
exp commands:		<b>O</b>	
<code>\exp_args:NNf</code> . . . . .	154	options:	
<code>\exp_args:NNx</code> . . . . .	87, 131, 188	<code>*</code> . . . . .	5
<code>\exp_args:No</code> . . . . .	165, 268	<code>+</code> . . . . .	4
<code>\exp_args:Nx</code> . . . . .	35	<code>&lt;code<sub>1</sub>&gt;</code> . . . . .	4
<code>\exp_last_unbraced:Nf</code> . . . . .	318, 332, 346, 368	<code>&lt;code<sub>2</sub>&gt;</code> . . . . .	5
<code>\exp_last_unbraced:NNf</code> . . . . .	302	<code>&lt;kv<sub>1</sub>&gt;</code> . . . . .	4
<code>\exp_not:N</code> . . . . .	71, 225, 235, 243, 251	<code>&lt;tl<sub>1</sub>&gt;</code> . . . . .	4
<code>\exp_not:n</code> . . . . .	209, 313	<code>&lt;tl<sub>2</sub>&gt;</code> . . . . .	4
<code>\expandafter</code> . . . . .	180	<code>&lt;tl<sub>3</sub>&gt;</code> . . . . .	5
<code>\ExplSyntaxOff</code> . . . . .	405	<code>&lt;tl<sub>4</sub>&gt;</code> . . . . .	5
<code>\ExplSyntaxOn</code> . . . . .	3	<code>&lt;tl<sub>5</sub>&gt;</code> . . . . .	5
<b>F</b>		<code>&lt;tl<sub>6</sub>&gt;</code> . . . . .	5
file commands:		<code>Expans</code> . . . . .	5
<code>\file_input:n</code> . . . . .	110	<code>File</code> . . . . .	6
<b>G</b>		<code>Inner</code> . . . . .	6
<code>\gappto</code> . . . . .	201	<code>Name</code> . . . . .	6
<b>I</b>		<code>Outer</code> . . . . .	6
<code>\IfBooleanT</code> . . . . .	195	<code>Separ</code> . . . . .	6
<code>\IfValueT</code> . . . . .	193, 207	<code>Write</code> . . . . .	6
<code>\IfValueTF</code> . . . . .	398	<b>P</b>	
int commands:		<code>\pdfcreationdate</code> . . . . .	314, 315
<code>\int_case:nnTF</code> . . . . .	55	prg commands:	
iow commands:		<code>\prg_replicate:nn</code> . . . . .	58, 89
<code>\iow_close:N</code> . . . . .	94, 98	prop commands:	
<code>\iow_new:N</code> . . . . .	93	<code>\prop_clear_new:N</code> . . . . .	268
<code>\iow_now:Nn</code> . . . . .	123	<code>\prop_gclear_new:N</code> . . . . .	46
<code>\iow_open:Nn</code> . . . . .	105	<code>\prop_gput:Nnn</code> . . . . .	29, 257
<code>\item</code> . . . . .	297, 298	<code>\prop_if_exist:NIF</code> . . . . .	277
<b>K</b>		<code>\prop_item:Nn</code> . . . . .	257, 281, 292
keys commands:		<code>\prop_map_function:NN</code> . . . . .	259
<code>\l_keys_choice_tl</code> . . . . .	310	<code>\prop_new:N</code> . . . . .	26, 286
<b>Q</b>		<code>\ProvideDocumentCommand</code> . . . . .	132, 386
quark commands:		<b>Q</b>	
<code>\q_stop</code> . . . . .	9, 15, 33, 42		

<b>S</b>			
seq commands:			
\seq_gclear_new:N	19, 82	\tl_count:n	78
\seq_gput_right:Nn	11, 292	\tl_gset:Nn	104, 221, 229, 239, 313
\seq_if_empty:NTF	47	\tl_gset_eq:NN	310
\seq_map_function:NN	20, 50, 146, 273, 294	\tl_log:n	111, 124
\seq_set_from_clist:Nn	166, 272	\tl_new:N	32, 101, 118
\seq_use:Nnnn	303	\tl_trim_spaces:n	11, 37, 38
<b>T</b>		<b>U</b>	
tl commands:		use commands:	
\c_empty_tl	48, 67, 163, 183, 185, 206	\use:N	36, 404
		\use_i:nn	61, 63
		\use_ii:nn	62
		\usepackage	4