

# The `luamml` package <sup>\*</sup>

Marcel Krüger

October 30, 2024

## 1 Use case

When generating output for the web or tagged output, mathematical content should often be represented as MathML. This uses Lua<sub>T</sub><sub>E</sub>X callbacks to automatically attempt to convert Lua<sub>T</sub><sub>E</sub>X math mode output into MathML.

## 2 Usage

The `luamml` package is designed to be used in automated ways by other packages and usually should not be invoked directly by the end user. For experiments, `luamml-demo` is included which provides easier to use interfaces.

Add in your preamble

```
\usepackage[files]{luamml-demo}
```

This will trigger the output of individual files for each block of math output containing corresponding MathML.

Alternatively

```
\usepackage[l3build]{luamml-demo}
```

will generate a single file with a concatenation of all MathML blocks.

For automated use, the `luamml` package can be included directly, followed by enclosing blocks which should generate files with `luamml_begin_single_file:` and `luamml_end_single_file:`. The filename can be set with `luamml_set_filename:n`.

## 3 Improving MathML conversion

When using constructs which do not automatically get converted in acceptable form, conversion hints can be provided with `luamml_annotate:en`. This allows to provide a replacement MathML structure in Lua table form, for example

```
\luamml_annotate:en {  
  nucleus = true,  
  core = {[0] = 'mi', 'TeX'},  
}{  
  \hbox{\TeX}  
}
```

---

<sup>\*</sup>This document corresponds to `luamml` v0.2.0, dated 2024-10-30.

produces a `<mi>TeX</mi>` element in the output instead of trying to import `TeX` as a mathematical expression. The table structure is explained in an appendix.

## 4 Features & Limitations

Currently all mathematical expressions which purely contain Unicode encoded math mode material without embedded non-math should get converted successfully. Usage with non-Unicode math (`TeX`'s 8-bit math fonts) is highly experimental and undocumented. Any attempt to build complicated structures by embedding arbitrary `TeX` code in the middle of math mode needs to have a MathML replacement specified. We try to automate more cases in the future.

## A Luamml's representation of XML and MathML

In the following I assume basic familiarity with both `LuaTeX`'s representation of math nodes and MathML.

### A.1 Representation of XML elements

In many places, `luamml` passes around XML elements. Every element is represented by a Lua table. Element 0 must always be present and is a string representing the tag name. The positive integer elements of the table represent child elements (either strings for direct text content or nested tables for nested elements). All string members which do not start with a colon are attributes, whose value is the result of applying `tostring` to the field value. This implies that these values should almost always be strings, except that the value 0 (since it never needs a unit) can often be set as a number. For example the XML document

```
<math block="display">
  <mn>0</mn>
  <mo> &lt; </mo>
  <mi mathvariant="normal">x</mi>
</math>
```

would be represented by the Lua table

```
{[0] = "math", block="display",
 {[0] = "mn", "0"},
 {[0] = "mo", "<"},
 {[0] = "mi", mathvariant="normal", "x"}
}
```

### A.2 Expression cores

MathML knows the concept of “embellished operators”: “The precise definition of an “embellished operator” is:

- an `<mo>` element;

- or one of the elements `<msub>`, `<msup>`, `<msubsup>`, `<munder>`, `<mover>`, `<munderover>`, `<mmultiscripts>`, `<mfrac>`, or `<semantics>` (§ 5.1 Annotation Framework), whose first argument exists and is an embellished operator;
- or one of the elements `<mstyle>`, `<mphantom>`, or `<mpadded>`, such that an `mrow` containing the same arguments would be an embellished operator;
- or an `<maction>` element whose selected sub-expression exists and is an embellished operator;
- or an `<mrow>` whose arguments consist (in any order) of one embellished operator and zero or more space-like elements.

For every embellished operator, MathML calls the `<mo>` element defining the embellished operator the “core” of the embellished operator.

Luamml makes this slightly more general: Every expression is represented by a pair of two elements: The expression and its core. The core is always a `<mo>`, `<mi>`, or `<mn>`, `nil` or `s` special marker for space like elements.

If and only if the element is an embellished operator the core is a `<mo>` element representing the core of the embellished operator. The core is a `<mi>` or a `<mn>` element if and only if the element would be an embellished operator with this core if this element were a `<mo>` element. The core is the special space like marker for space like elements. Otherwise the core is `nil`.

### A.3 Translation of math noads

A math lists can contain the following node types: `noad`, `fence`, `fraction`, `radical`, `accent`, `style`, `choice`, `ins`, `mark`, `adjust`, `boundary`, `whatsit`, `penalty`, `disc`, `glue`, and `kern`. The “noads”

#### A.3.1 Translation of kernel noads

The math noads of this list contain nested kernel noads. So in the first step, we look into how kernel nodes are translated to math nodes.

**math\_char kernel noads** First the family and character value in the `math_char` are used to lookup the Unicode character value of this `math_char`. (For `unicode-math`, this is usually just the character value. Legacy maths has to be remapped based on the family.) Then there are two cases: The digits 0 to 9 are mapped to `<mn>` elements, everything else becomes a `<mi>` element with `mathvariant` set to `normal`. (The `mathvariant` value might get suppressed if the character defaults to `mathvariant normal`.) In either case, the `tex:family` attribute is set to the family number if it’s not 0.

The core is always set to the expression itself. E.g. the `math_char` kernel noad `\fam3 a` would become (assuming no remapping for this family)

```
{[0] = 'mi',
  mathvariant = 'normal',
  ["tex:family"] = 3,
  "a"
}
```

### A.3.2 sub\_box kernel noads

I am open to suggestions how to convert them properly.

### A.3.3 sub\_mlist kernel noads

The inner list is converted as a `<mrow>` element, with the core being the core of the `<mrow>` element. See the rules for this later.

### A.3.4 delim kernel noads

If the `small_char` is zero, these get converted as space like elements of the form

```
{[0] = 'mspace',  
  width = '1.196pt',  
}
```

where 1.196 is replaced by the current value of `\nulldelimiterspace` converted to bp.

Otherwise the same rules as for `math_char` apply, except that instead of `mi` or `<mn>` elements, no elements are generated, `mathvariant` is never set, `stretchy` is set to `true` if the operator is not on the list of default stretchy operators in the MathML specification and `lspace` and `rspace` attributes are set to zero.

### A.3.5 acc kernel noads

Depending on the surrounding element containing the `acc` kernel noad, it is either stretchy or not. If it's stretchy, the same rules as for `delim` apply, except that `lspace` and `rspace` are not set. Otherwise the `stretchy` attribute is set to false if the operator is on the list of default stretchy operators.

## B Package Implementation

### B.1 Initialization

```
1 <@@=luamml>  
2 <*luatex>  
3 \ProvidesExplPackage {luamml} {2024-10-30} {0.2.0}  
4   {Automatically generate presentational MathML from LuaTeX math expressions}  
5 </luatex>  
6 <*pdfTeX>  
7 \ProvidesExplPackage {luamml-pdf} {2024-10-30} {0.2.0}  
8   {MathML generation for Lua pdfLaTeX}  
9 </pdfTeX>
```

### B.2 Initialization

These variable have to appear before the Lua module is loaded and will be used to communicate information to the callback.

Here `\tracingmathml` does not use a `expl3` name since it is not intended for programming use but only as a debugging helper for the user. The other variables are internal, but we provide public interfaces for setting them later.

```
10 \int_new:N \l__luamml_flag_int  
11 \int_new:N \l__luamml_pretty_int
```

```

12 <luatex>\tl_new:N \l__luamml_filename_tl
13 \tl_new:N \l__luamml_root_tl
14 \tl_set:Nn \l__luamml_root_tl { mrow }
15 \tl_new:N \l__luamml_label_tl
16 <pdfTeX>\int_new:N \g__luamml_formula_id_int
17 <luatex>\int_new:N \tracingmathml
18
19 \int_set:Nn \l__luamml_pretty_int { 1 }

```

Now we can load the Lua module which defines the callback. Of course until pdfTeX starts implementing `\directlua` this is only done in LuaTeX.

```

20 <luatex>\lua_now:n { require'luamml-tex' }

```

### B.3 Hook

We also call a hook with arguments at the end of every MathML conversion with the result. Currently only implemented in LuaTeX since it immediately provides the output.

```

21 <*luatex>
22 \hook_new_with_args:nn { luamml / converted } { 1 }
23
24 \cs_new_protected:Npn \__luamml_output_hook:n {
25   \hook_use:nw { luamml / converted } { 1 }
26 }
27 \__luamml_register_output_hook:N \__luamml_output_hook:n
28 </luatex>

```

### B.4 Flags

The most important interface is for setting the flag which controls how the formulas should be converted.

`\luamml_process:` Consider the current formula to be a complete, free-standing mathematical expression which should be converted to MathML. Additionally, the formula is also saved in the `start_math` node as with `\luamml_save:.`

```

29 \cs_new_protected:Npn \luamml_process: {
30   \tl_set:Nn \l__luamml_label_tl {}
31   \int_set:Nn \l__luamml_flag_int { 3 }
32 }

```

Temporarily for compatibility

```

33 \cs_set_eq:NN \luamml_flag_process: \luamml_process:

```

*(End of definition for `\luamml_process:.` This function is documented on page ??.)*

`\__luamml_maybe_structelem:` A internal helper which can be added to a tag to preserve the external state of the `structelem` flag.

```

34 \cs_new:Npn \__luamml_maybe_structelem: {
35   (
36     8 * \int_mod:nn {
37       \int_div_truncate:nn { \l__luamml_flag_int } {8}
38     } {2}
39   ) +
40 }

```

(End of definition for `\__luamml_maybe_structelem:`)

`\__luamml_style_to_num:N`

```
41 \cs_new:Npn \__luamml_style_to_num:N #1 {
42   \luatex 32 * #1
43   \pdfTeX
44   \token_case_meaning:NnF #1 {
45     \displaystyle {0}
46     \textstyle {32}
47     \scriptstyle {64}
48     \scriptscriptstyle {96}
49   } {
50     \Invalid_mathstyle
51   }
52 \end{pdfTeX}
53 }
```

(End of definition for `\__luamml_style_to_num:N`.)

`\luamml_save:n` Convert the current formula but only save its representation in the math node without emitting it as a complete formula. This is useful when the expression forms part of a bigger formula and will be integrated into its MathML tables later by special code. It optionally accepts three parameters: A label, one math style command (`\displaystyle`, `\textstyle`, etc.) which is the implicit math style (so the style which the surrounding code expects this style to have) and a name for the root element (defaults to `mrow`). If the root element name is `mrow`, it will get suppressed in some cases.

```
54 \cs_new_protected:Npn \luamml_save:n #1 {
55   \tl_set:Nn \l__luamml_label_tl {#1}
56   \int_set:Nn \l__luamml_flag_int { \__luamml_maybe_structelem: 1 }
57 }
58 \cs_new_protected:Npn \luamml_save:nN #1#2 {
59   \tl_set:Nn \l__luamml_label_tl {#1}
60   \int_set:Nn \l__luamml_flag_int { \__luamml_maybe_structelem: 17 + \__luamml_style_to_num:N #2 }
61 }
62 \cs_new_protected:Npn \luamml_save:nn #1 {
63   \tl_set:Nn \l__luamml_label_tl {#1}
64   \int_set:Nn \l__luamml_flag_int { \__luamml_maybe_structelem: 5 }
65   \tl_set:Nn \l__luamml_root_tl
66 }
67 \cs_new_protected:Npn \luamml_save:nNn #1#2 {
68   \tl_set:Nn \l__luamml_label_tl {#1}
69   \int_set:Nn \l__luamml_flag_int { \__luamml_maybe_structelem: 21 + \__luamml_style_to_num:N #2 }
70   \tl_set:Nn \l__luamml_root_tl
71 }
```

Temporarily for compatibility

```
72 \cs_set_eq:NN \luamml_flag_save:n \luamml_save:n
73 \cs_set_eq:NN \luamml_flag_save:nN \luamml_save:nN
74 \cs_set_eq:NN \luamml_flag_save:nn \luamml_save:nn
75 \cs_set_eq:NN \luamml_flag_save:nNn \luamml_save:nNn
```

(End of definition for `\luamml_save:n` and others. These functions are documented on page ??.)

`\luamml_ignore:` Completely ignore the math mode material.

```
76 \cs_new_protected:Npn \luamml_ignore: {  
77   \int_set:Nn \l__luamml_flag_int { 0 }  
78 }
```

Temporarily for compatibility

```
79 \cs_set_eq:NN \luamml_flag_ignore: \luamml_ignore:
```

*(End of definition for \luamml\_ignore:. This function is documented on page ??.)*

`\luamml_structelem:` Like `\luamml_process:`, but additionally adds PDF structure elements. This only works in LuaTeX and requires that the `tagpdf` package has been loaded *before* `luamml`.

```
80 <*luatex>  
81 \cs_new_protected:Npn \luamml_structelem: {  
82   \tl_set:Nn \l__luamml_label_tl {}  
83   \int_set:Nn \l__luamml_flag_int { 11 }  
84 }
```

Temporarily for compatibility

```
85 \cs_set_eq:NN \luamml_flag_structelem: \luamml_structelem:  
86 </luatex>
```

*(End of definition for \luamml\_structelem:. This function is documented on page ??.)*

`\luamml_set_filename:n` Allows to set a filename to which the generated MathML gets written. Previous content from the file will get overwritten. This includes results written by a previous formula. Therefore this has to be called separately for every formula or it must expand to different values to be useful. The value is fully expanded when the file is written.

Only complete formulas get written into files (so formulas where `\luamml_process:` or `\luamml_structelem:` are in effect).

Only implemented in LuaTeX, in pdfTeX the arguments for `pdfmml` determine the output location.

```
87 <*luatex>  
88 \cs_new_protected:Npn \luamml_set_filename:n {  
89   \tl_set:Nn \l__luamml_filename_tl  
90 }  
91 </luatex>
```

*(End of definition for \luamml\_set\_filename:n. This function is documented on page ??.)*

`\luamml_begin_single_file:` Everything between these two commands gets written into the same XML file. The  
`\luamml_end_single_file:` filename is expanded when `\luamml_begin_single_file:` gets executed.

(Implemented in Lua)

*(End of definition for \luamml\_begin\_single\_file: and \luamml\_end\_single\_file:. These functions are documented on page ??.)*

By default, the flag is set to assume complete formulas.

```
92 \luamml_process:
```

## B.5 Annotations

These are implemented very differently depending on the engine, but the interface should be the same.

### B.5.1 LuaTeX

93 `<*luatex>`

`\luamml_annotate:nen` A simple annotation scheme: The first argument is the number of top level noads to be annotated, the second parameter the annotation and the third parameter the actual list of math tokens. The first argument can be omitted to let LuaTeX determine the number itself.

`\luamml_annotate:en`

Passing the first parameter explicitly is useful for any annotations which should be compatible with future pdfTeX versions of this functionality.

```
94 \cs_new_protected:Npn \luamml_annotate:nen #1#2#3 {
95   \__luamml_annotate_begin:
96     #3
97   \__luamml_annotate_end:we \tex_numexpr:D #1 \scan_stop: {#2}
98 }
99
100 \cs_new_protected:Npn \luamml_annotate:en #1#2 {
101   \__luamml_annotate_begin:
102     #2
103   \__luamml_annotate_end:e {#1}
104 }
```

*(End of definition for `\luamml_annotate:nen` and `\luamml_annotate:en`. These functions are documented on page ??.)*

105 `</luatex>`

### B.5.2 pdfTeX

106 `<*pdfTeX>`

`\__luamml_pdf_showlists:` Here and in many other locations the pdfTeX implementation is based on `\showlists`, so we define an internal wrapper which sets all relevant parameters.

```
107 \cs_if_exist:NTF \showstream {
108   \iow_new:N \l__luamml_pdf_stream
109   \iow_open:Nn \l__luamml_pdf_stream { \jobname .tml }
110   \cs_new_protected:Npn \__luamml_pdf_showlists: {
111     \group_begin:
112       \int_set:Nn \tex_showboxdepth:D { \c_max_int }
113       \int_set:Nn \tex_showboxbreadth:D { \c_max_int }
114       \showstream = \l__luamml_pdf_stream
115       \tex_showlists:D
116     \group_end:
117   }
118 } {
119   \cs_set_eq:NN \l__luamml_pdf_stream \c_log_iow
120   \cs_set_eq:NN \__luamml_pdf_set_showstream: \scan_stop:
121   \cs_new_protected:Npn \__luamml_pdf_showlists: {
122     \group_begin:
123       \int_set:Nn \l_tmpa_int { \tex_interactionmode:D }
124       \int_set:Nn \tex_interactionmode:D { 0 }
125       \int_set:Nn \tex_showboxdepth:D { \c_max_int }
126       \int_set:Nn \tex_showboxbreadth:D { \c_max_int }
127       \tex_showlists:D
128       \int_set:Nn \tex_interactionmode:D { \l_tmpa_int }
129     \group_end:
```



```

130 }
131 }

```

(End of definition for `\_luamml_pdf_showlists:`.)

`\luamml_annotate:n` Now we can define the annotation commands for pdfTeX.

```

\luamml_annotate:en
\luamml_annotate:en
132 \cs_generate_variant:Nn \tl_to_str:n { e }
133 \int_new:N \g__luamml_annotation_id_int
134 \cs_new_protected:Npn \luamml_annotate:nen #1#2#3 {
135   \int_gincr:N \g__luamml_annotation_id_int
136   \iow_shipout_x:Nx \l__luamml_pdf_stream {
137     LUAMML_MARK_REF:
138     \int_use:N \g__luamml_annotation_id_int
139     :
140   }
141   \iow_now:Nx \l__luamml_pdf_stream {
142     LUAMML_MARK:
143     \int_use:N \g__luamml_annotation_id_int
144     :
145     count = \int_eval:n {#1},
146     #2
147     \iow_newline:
148     LUAMML_MARK_END
149   }
150   #3
151 }
152 \cs_new_protected:Npn \luamml_annotate:en #1#2 {
153   \int_gincr:N \g__luamml_annotation_id_int
154   \iow_shipout_x:Nx \l__luamml_pdf_stream {
155     LUAMML_MARK_REF:
156     \int_use:N \g__luamml_annotation_id_int
157     :
158   }
159   \iow_now:Nx \l__luamml_pdf_stream {
160     LUAMML_MARK:
161     \int_use:N \g__luamml_annotation_id_int
162     :
163     count = data.count[\int_use:N \g__luamml_annotation_id_int],
164     #1
165     \iow_newline:
166     LUAMML_MARK_END
167   }
168   \use:x {
169     \iow_now:Nn \l__luamml_pdf_stream {
170       LUAMML_COUNT:
171       \int_use:N \g__luamml_annotation_id_int
172     }
173     \_luamml_pdf_showlists:
174     \exp_not:n {#2}
175     \iow_now:Nn \l__luamml_pdf_stream {
176       LUAMML_COUNT_END:
177       \int_use:N \g__luamml_annotation_id_int
178     }
179     \_luamml_pdf_showlists:

```

```

180 }
181 }

```

(End of definition for `\luamml_annotate:nen` and `\luamml_annotate:en`. These functions are documented on page ??.)

```

182 </pdfTeX>

```

## B.6 Trigger for specific formula

This only applies for pdfTeX since in LuaTeX everything is controlled by the callback, but for compatibility the function is defined anyway.

`\luamml_pdf_write:` We could accept parameters for the flag and tag here, but for compatibility with LuaTeX they are passed in macros instead.

```

183 <*pdfTeX>
184 \cs_new_protected:Npn \luamml_pdf_write: {
185   \int_gincr:N \g__luamml_formula_id_int
186   \iow_now:Nx \l__luamml_pdf_stream {
187     LUAMML_FORMULA_BEGIN:
188     \int_use:N \g__luamml_formula_id_int
189     :
190     \int_use:N \l__luamml_flag_int
191     :
192     \l__luamml_root_tl
193     :
194     \l__luamml_label_tl
195   }
196   \__luamml_pdf_showlists:
197   \iow_now:Nx \l__luamml_pdf_stream {
198     LUAMML_FORMULA_END
199   }
200 }
201 </pdfTeX>
202 <luaTeX>\cs_new_eq:NN \luamml_pdf_write: \scan_stop:

```

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

## B.7 Further helpers

`\RegisterFamilyMapping` The Lua version of this is defined in the Lua module.

```

203 <*pdfTeX>
204 \NewDocumentCommand \RegisterFamilyMapping {m m} {
205   \iow_now:Nx \l__luamml_pdf_stream {
206     LUAMML_INSTRUCTION:REGISTER_MAPPING: \int_use:N #1 : #2
207   }
208 }
209 </pdfTeX>

```

(End of definition for `\RegisterFamilyMapping`. This function is documented on page ??.)

## B.8 Patching

For some packages, we ship with patches to make them more compatible and to demonstrate how other code can be patched to work with `luamml`.

These are either loaded directly if the packages are loaded or delayed using L<sup>A</sup>T<sub>E</sub>X's hook system otherwise.

`__luamml_patch_package:nn` For this, we use two helpers: First a wrapper which runs arbitrary code either now (if the package is already loaded) or as soon as the package loads, second an application of the first one to load packages following `luamml`'s naming scheme for these patch packages.

```

210 \cs_new_protected:Npn __luamml_patch_package:nn #1 #2 {
211   \@ifpackageloaded {#1} {#2} {
212     \hook_gput_code:nnn {package/#1/after} {luamml} {#2}
213   }
214 }
215 \cs_new_protected:Npn __luamml_patch_package:n #1 {
216   __luamml_patch_package:nn {#1} {
217     \RequirePackage { luamml-patches-#1 }
218   }
219 }

```

(End of definition for `__luamml_patch_package:nn` and `__luamml_patch_package:n`.)

We currently provide minimal patching for the kernel, `amsmath` and `array`. Currently only the kernel code supports pdf<sub>T</sub>E<sub>X</sub>, but it's planned to extend this.

```

220 \RequirePackage { luamml-patches-kernel }
221 \*luatex
222 __luamml_patch_package:n {amstext}
223 __luamml_patch_package:n {amsmath}
224 __luamml_patch_package:n {mathtools}
225 __luamml_patch_package:n {array}
226 \*luatex

```

## 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>C</b>                                       |  | <b>E</b>                                      |          |
| cs commands:                                   |  | exp commands:                                 |          |
| <code>\cs_generate_variant:Nn</code> . . . . . | 132  | <code>\exp_not:n</code> . . . . .             | 174      |
| <code>\cs_if_exist:NTF</code> . . . . .        | 107  |   |          |
| <code>\cs_new:Npn</code> . . . . .             | 34, 41                                     | <b>G</b>                                      |          |
| <code>\cs_new_eq:NN</code> . . . . .           | 202  | group commands:                               |          |
| <code>\cs_new_protected:Npn</code> . . . . .   | 24,  | <code>\group_begin:</code> . . . . .          | 111, 122 |
|  | 29, 54, 58, 62, 67, 76, 81, 88, 94,        | <code>\group_end:</code> . . . . .            | 116, 129 |
|  | 100, 110, 121, 134, 152, 184, 210, 215     |   |          |
| <code>\cs_set_eq:NN</code> . . . . .           |  | <b>H</b>                                      |          |
|  | . . . 33, 72, 73, 74, 75, 79, 85, 119, 120 | hook commands:                                |          |
| <b>D</b>                                       |  | <code>\hook_gput_code:nnn</code> . . . . .    | 212      |
| <code>\directlua</code> . . . . .              | 5  | <code>\hook_new_with_args:nn</code> . . . . . | 22       |
| <code>\displaystyle</code> . . . . .           | 6, 45                                      | <code>\hook_use:nnw</code> . . . . .          | 25       |

|   |  |
|---|--|
| <b>I</b>  |  |
| int commands:   |  |
| \int_div_truncate:nn  | 37   |
| \int_eval:n   | 145  |
| \int_gincr:N  | 135, 153, 185  |
| \int_mod:nn   | 36   |
| \int_new:N  | 10, 11, 16, 17, 133  |
| \int_set:Nn   | 19, 31, 56, 60, 64, 69,<br>77, 83, 112, 113, 123, 124, 125, 126, 128 |
| \int_use:N  | 138, 143,<br>156, 161, 163, 171, 177, 188, 190, 206                  |
| \c_max_int  | 112, 113, 125, 126   |
| \l_tmpa_int   | 123, 128   |
| Invalid commands:   |  |
| \Invalid_mathstyle  | 50   |
| iow commands:   |  |
| \iow_new:N  | 108  |
| \iow_newline:   | 147, 165   |
| \iow_now:Nn   | 141, 159, 169, 175, 186, 197, 205                                    |
| \iow_open:Nn  | 109  |
| \iow_shipout_x:Nn   | 136, 154   |
| \c_log_iow  | 119  |
| <b>J</b>  |  |
| \jobname  | 109  |
| <b>L</b>  |  |
| lua commands:   |  |
| \lua_now:n  | 20   |
| luamml commands:  |  |
| \luamml_annotate:nn   | 94, 100, 132, 152  |
| luamml_annotate:nn  | 1  |
| \luamml_annotate:nnn  | 94, 94, 132, 134   |
| \luamml_begin_single_file:  | 7, 92  |
| luamml_begin_single_file:   | 1  |
| \luamml_end_single_file:  | 92   |
| luamml_end_single_file:   | 1  |
| \luamml_flag_ignore:  | 79   |
| \luamml_flag_process:   | 33   |
| \luamml_flag_save:n   | 72   |
| \luamml_flag_save:nN  | 73   |
| \luamml_flag_save:nn  | 74   |
| \luamml_flag_save:nNn   | 75   |
| \luamml_flag_structelem:  | 85   |
| \luamml_ignore:   | 76, 76, 79   |
| \luamml_pdf_write:  | 183, 184, 202  |
| \luamml_process:  | 7, 29, 29, 33, 92  |
| \luamml_save:   | 5  |
| \luamml_save:n  | 54, 54, 72   |
| \luamml_save:nN   | 54, 58, 73   |
| \luamml_save:nn   | 54, 62, 74   |
| \luamml_save:nNn  | 54, 67, 75   |
| \luamml_set_filename:n  | 87, 88   |
| luamml_set_filename:n   | 1  |
| \luamml_structelem:   | 7, 80, 81, 85  |
| luamml internal commands:   |  |
| \_luamml_annotate_begin:  | 95, 101  |
| \_luamml_annotate_end:n   | 103  |
| \_luamml_annotate_end:we  | 97   |
| \g_luamml_annotation_id_int   | 133, 135,<br>138, 143, 153, 156, 161, 163, 171, 177                  |
| \l_luamml_filename_tl   | 12, 89   |
| \l_luamml_flag_int  | 10,<br>31, 37, 56, 60, 64, 69, 77, 83, 190                           |
| \g_luamml_formula_id_int  | 16, 185, 188   |
| \l_luamml_label_tl  | 15, 30, 55, 59, 63, 68, 82, 194                                      |
| \_luamml_maybe_structelem:  | 34, 34, 56, 60, 64, 69   |
| \_luamml_output_hook:n  | 24, 27   |
| \_luamml_patch_package:n  | 210, 215, 222, 223, 224, 225   |
| \_luamml_patch_package:nn   | 210, 210, 216  |
| \_luamml_pdf_set_showstream:  | 120  |
| \_luamml_pdf_showlists:   | 107, 110, 121, 173, 179, 196   |
| \l_luamml_pdf_stream  | 108, 109, 114, 119, 136,<br>141, 154, 159, 169, 175, 186, 197, 205   |
| \l_luamml_pretty_int  | 11, 19   |
| \_luamml_register_output_hook:N   | 27   |
| \l_luamml_root_tl   | 13, 14, 65, 70, 192  |
| \_luamml_style_to_num:N   | 41, 41, 60, 69   |
| <b>N</b>  |  |
| \NewDocumentCommand   | 204  |
| <b>P</b>  |  |
| \ProvidesExplPackage  | 3, 7   |
| <b>R</b>  |  |
| \RegisterFamilyMapping  | 203  |
| \RequirePackage   | 217, 220   |
| <b>S</b>  |  |
| scan commands:  |  |
| \scan_stop:   | 97, 120, 202   |
| \scriptscriptstyle  | 48   |
| \scriptstyle  | 47   |
| \showlists  | 8  |
| \showstream   | 107, 114   |
| <b>T</b>  |  |
| T <sub>E</sub> X and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands: |  |
| \ifpackageloaded  | 211  |

