

stex.sty: \TeX 2.0*

Michael Kohlhase, Dennis Müller
FAU Erlangen-Nürnberg
<http://kwarc.info/>

2021-09-03

Abstract

TODO

1 Introduction

TODO

*Version v1.9 (last revised 2021/08/01)

Contents

1	Introduction	1
2	Manual	3
2.1	Modules	3
2.2	Semantic Macros and Notations	3
2.3	Archives and Imports	7
3	Documentation	8
3.1	Utils	8
3.2	Files, Paths, URIs	9
3.3	MathHub Archives	10
3.4	The Module System	12
3.5	Symbols and Terms	20
4	Implementation	25
4.1	The \LaTeX document class	25
4.2	Preliminaries	25
4.3	Files, Paths and URIs	30
4.4	MathHub Repositories	33
4.5	Module System	37
4.6	Symbol Declarations	53
4.7	Notations	59
4.8	Terms	67
4.9	Auxiliary Packages	73

2 Manual

2.1 Modules

`{module}`, `{@module}`

2.2 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

To declare a symbol (in a module), we use `\symdecl`, which takes as argument the name of the corresponding semantic macro, e.g. `\symdecl{foo}` introduces the macro `\foo`. Additionally, `\symdecl` takes several options, the most important one being its arity. `foo` as declared above yields a *constant* symbol. To introduce an *operator* which takes arguments, we have to specify which arguments it takes.

For example, to introduce binary multiplication, we can do `\symdecl[args=2]{mult}`. We can then supply the semantic macro with arbitrarily many notations, such as `\notation{mult}{#1 #2}`.

Example 1

```
\symdecl[args=2]{mult}
\notation{mult}{#1 #2}
 $\mult{a}{b}$ 
```

ab

Since usually, a freshly introduced symbol also comes with a notation from the start, the `\symdef` command combines `\symdecl` and `\notation`. So instead of the above, we could have also written

```
\symdef[args=2]{mult}{#1 #2}
```

Adding more notations like `\notation[cdot]{mult}{#1 \comp{\cdot} #2}` or `\notation[times]{mult}{#1 \comp{\times} #2}` allows us to write $\mult[cdot]{a}{b}$ and $\mult[times]{a}{b}$:

Example 2

```
\notation[cdot]{mult}{#1 \comp{\cdot} #2}
\notation[times]{mult}{#1 \comp{\times} #2}
 $\mult[cdot]{a}{b}$  and  $\mult[times]{a}{b}$ 
```

$a \cdot b$ and $a \times b$

Not using an explicit option with a semantic macro yields the first declared notation, unless changed¹.

¹EdNOTE: TODO

Outside of math mode, or by using the starred variant `\foo*`, allows to provide a custom notation, where notational (or textual) components can be given explicitly in square brackets.

Example 3

```
$\mult*{a}[\comp{\ast}]{b}$ is the
\mult[\comp{product of}]{a}[\comp{and}]{b}
```

$a*b$ is the *product of* a and b

In custom mode, prefixing an argument with a star will not print that argument, but still export it to OMDoc:

Example 4

```
\mult[\comp{Multiplying}]*{$\mult{a}{b}$} again by {b$} yields...
```

Multiplying again by b yields...

The syntax `*[int]` allows switching the order of arguments. For example, given a 2-ary semantic macro `\forevery` with exemplary notation `\forall #1. #2`, we can write

Example 5

```
\symdecl[args=2]{forevery}
\forevery*[2]{The proposition $P$}[\comp{holds for every}]*[1]{x\in A}
```

The proposition P holds for every $x \in A$

When using `*[n]`, after reading the provided (n th) argument, the “argument counter” automatically continues where we left off, so the `*[1]` in the above example can be omitted.

For a macro with arity > 0 , we can refer to the operator *itself* semantically by suffixing the semantic macro with an exclamation point `!` in either text or math mode.

Example 6

```
\mult![\comp{Multiplication}] (denoted by $\mult![\comp{cdot}]$) is defined by...
```

Multiplication (denoted by \cdot) is defined by...

The macro `\comp` as used everywhere above is responsible for highlighting, linking, and tooltips, and should be wrapped around the notation (or text) components that should be treated accordingly. While it is attractive to just wrap a whole notation, this would also wrap around e.g. the arguments themselves, so instead, the user is tasked with marking the notation components themselves.

The precise behaviour of `\comp` is governed by the macro `\@comp`, which takes two arguments: The tex code of the text (unexpanded) to highlight, and the URI of the current symbol. `\@comp` can be safely redefined to customize the behaviour.

The starred variant `\symdecl*{foo}` does not introduce a semantic macro, but still declares a corresponding symbol. `foo` (like any other symbol, for that matter) can then be accessed via `\STEXsymbol{foo}` or (if `foo` was declared in a module `Foo`) via `\STEXModule{Foo}?{foo}`.

both `\STEXsymbol` and `\STEXModule` take any arbitrary ending segment of a full URI to determine which symbol or module is meant. e.g. `\STEXsymbol{Foo?foo}` is also valid, as are e.g. `\STEXModule{path?Foo}?{foo}` or `\STEXsymbol{path?Foo?foo}`

2.2.1 Other Argument Types

So far, we have stated the arity of a semantic macro directly. This works if we only have “normal” (or more precisely: i-type) arguments. To make use of other argument types, instead of providing the arity numerically, we can provide it as a sequence of characters representing the argument types – e.g. instead of writing `args=2`, we can equivalently write `args=ii`, indicating that the macro takes two i-type arguments.

Besides i-type arguments, \TeX has two other types, which we will discuss now.

The first are *binding* (b-type) arguments, representing variables that are *bound* by the operator. This is the case for example in the above `\forevery`-macro: The first argument is not actually an argument that the `forevery` “function” is “applied” to; rather, the first argument is a new variable (e.g. x) that is *bound* in the subsequent argument. More accurately, the macro should therefore have been implemented thusly:

```
\symdef[args=bi]{forevery}{\forall #1.\; #2}
```

b-type arguments are indistinguishable from i-type arguments within \TeX , but are treated very differently in OMDoc and by MMT. More interesting *within* \TeX are a-type arguments, which represent (associative) arguments of flexible arity, which are provided as comma-separated lists. This allows e.g. better representing the `\mult`-macro above:

Example 7

```
\symdef[ args=a]{mult}{#1}{#1 \comp \cdot #2}
 $\mult{a,b,c,\{d^e\},f}$ 
```

 $a \cdot b \cdot c \cdot d^e \cdot f$

As the example above shows, notations get a little more complicated for associative arguments. For every a-type argument, the `\notation`-macro takes an additional argument that declares how individual entries in an a-type argument list are aggregated. The first notation argument then describes how the aggregated expression is combined into the full representation.

For a more interesting example, consider a flexary operator for ordered sequences in ordered set, that taking arguments $\{a, b, c\}$ and \mathbb{R} prints $a \leq b \leq c \in \mathbb{R}$. This operator takes two arguments (an a-type argument and an i-type argument), aggregates the individuals of the associative argument using \leq , and combines the result with \in and the second argument thusly:

Example 8

```
\symdef[ args=ai]{numseq}{#1 \comp\in #2}{#1 \comp\leq #2}
$\numseq{a,b,c}{\mathbb R}$
```

$$a \leq b \leq c \in \mathbb{R}$$

·2 3 4

2.2.2 Precedences

Every notation has an (upwards) *operator precedence* and for each argument a (downwards) *argument precedence* used for automated bracketing. For example, a notation for a binary operator $\texttt{\backslashfoo}$ could be declared like this:

```
\notation[prec=200;500x600]{foo}{#1 \comp{+} #2}
```

assigning an operator precedence of 200, an argument precedence of 500 for the first argument, and an argument precedence of 600 for the second argument.

TeX insert brackets thusly: Upon encountering a semantic macro (such as $\texttt{\backslashfoo}$), its operator precedence (e.g. 200) is compared to the current downwards precedence (initially $\texttt{\backslashneginfprec}$). If the operator precedence is *smaller* than the current downwards precedence, parentheses are inserted around the semantic macro.

Notations for symbols of arity 0 have a default precedence of $\texttt{\backslashinfprec}$, i.e. by default, parentheses are never inserted around constants. Notations for symbols with arity > 0 have a default operator precedence of 0. If no argument precedences are explicitly provided, then by default they are equal to the operator precedence.

Consequently, if some operator A should bind stronger than some operator B , then A operator precedence should be larger than B ’s argument precedences.

For example:

Example 9

```
\notation[prec=50]{plus}{#1 \comp{+} #2}
\notation[prec=100]{times}{#1 \comp{\cdot} #2}
$\plus{a}{\times{b}{c}}$ and $\times{a}{\plus{b}{c}}$
```

$$a + b \cdot c \text{ and } a \cdot (b + c)$$

²EdNOTE: what about e.g. $\int_x \int_y \int_z f \, dx \, dy \, dz$?

³EdNOTE: “decompose” a-type arguments into fixed-arity operators?

⁴EdNOTE: flexary b-type arguments (e.g. for all)?

2.3 Archives and Imports

2.3.1 Namespaces

Ideally, \S\TeX would use arbitrary URIs for modules, with no forced relationships between the *logical* namespace of a module and the *physical* location of the file declaring the module – like MMT does things.

Unfortunately, \TeX only provides very restricted access to the file system, so we are forced to generate namespaces systematically in such a way that they reflect the physical location of the associated files, so that \S\TeX can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completeness sake, we describe how they are constructed:

- If $\text{\begin{module}\{Foo\}}$ occurs in a file `/path/to/file/Foo[.<lang>].tex` which does not belong to an archive, the namespace is `file://path/to/file`.
- If the same statement occurs in a file `/path/to/file/bar[.<lang>].tex`, the namespace is `file://path/to/file/bar`.

In other words: outside of archives, the namespace corresponds to the file URI with the filename dropped iff it is equal to the module name, and ignoring the (optional) language suffix¹.

If the current file is in an archive, the procedure is the same except that the initial segment of the file path up to the archive's `source`-folder is replaced by the archive's namespace URI.

2.3.2 Paths in Import-Statements

Conversely, here is how namespaces/URIs and file paths are computed in import statements, exemplary \importmodule :

- $\text{\importmodule}\{Foo\}$ outside of an archive refers to module `Foo` in the current namespace. Consequently, `Foo` must have been declared earlier in the same document or, if not, in a file `Foo[.<lang>].tex` in the same directory.
- The same statement *within* an archive refers to either the module `Foo` declared earlier in the same document, or otherwise to the module `Foo` in the archive's top-level namespace. In the latter case, it has to be declared in a file `Foo[.<lang>].tex` directly in the archive's `source`-folder.
- Similarly, in $\text{\importmodule}\{some/path?Foo\}$ the path `some/path` refers to either the sub-directory and relative namespace path of the current directory and namespace outside of an archive, or relative to the current archive's top-level namespace and `source`-folder, respectively.

The module `Foo` must either be declared in the file `<top-directory>/some/path/Foo[.<lang>].tex`, or in `<top-directory>/some/path[.<lang>].tex` (which are checked in that order).

- Similarly, $\text{\importmodule}[Some/Archive]\{some/path?Foo\}$ is resolved like the previous cases, but relative to the archive `Some/Archive` in the mathhub-directory.

¹which is internally attached to the module name instead, but a user need not worry about that.

- Finally, `\importmodule{full://uri?Foo}` naturally refers to the module `Foo` in the namespace `full://uri`. Since the file this module is declared in can not be determined directly from the URI, the module must be in memory already, e.g. by being referenced earlier in the same document.

Since this is less compatible with a modular development, using full URIs directly is discouraged.

3 Documentation

3.1 Utils

<code>\sTeX</code>	both print this sTeX logo.
<code>\stex</code>	

<code>\stex_debug:n</code>	<code>\stex_debug:n {⟨message⟩}</code>
----------------------------	----------------------------------------

Logs `⟨message⟩`, if the package option `debug` is used.

<code>\stex_kpsewhich:n</code>	<code>\stex_kpsewhich:n</code> executes <code>kpsewhich</code> and stores the return in <code>\l_stex_kpsewhich_return_str</code> . This does not require shell escaping.
--------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<code>\stex_addtosms:n</code>	Adds the provided code to the <code>.sms</code> -file of the document.
-------------------------------	------------------------------------------------------------------------

3.1.1 S_{CA}TeX, L_{AT}EXML and H_TML Annotations

<code>\if@latexml</code>	L _{AT} E _X 2e and L _{AT} E _X 3 conditionals for L _{AT} EXML.
<code>\latexml_if_p:</code>	
<code>\latexml_if:T</code>	
<code>\latexml_if:F</code>	
<code>\latexml_if:TF</code>	

We have four macros for annotating generated HTML (via L_{AT}EXML or S_{CA}TeX) with attributes:

<code>\stex_annotate:nnn</code>	<code>\stex_annotate:nnn {⟨property⟩} {⟨resource⟩} {⟨content⟩}</code>
<code>\stex_annotate_invisible:nnn</code>	
<code>\stex_annotate_invisible:n</code>	

Annotates the HTML generated by `⟨content⟩` with

`property="stex:⟨property⟩", resource="⟨resource⟩".`

`\stex_annotate_invisible:n` adds the attributes

`stex:visible="false", style="display:none".`

`\stex_annotate_invisible:nnn` combines the functionality of both.

<code>stex_annotate_env</code>	$\begin{array}{l} \backslash\text{begin}\{\text{stex_annotate_env}\}\{\langle\text{property}\rangle\}\{\langle\text{resource}\rangle\} \\ \langle\text{content}\rangle \\ \backslash\text{end}\{\text{stex_annotate_env}\} \end{array}$ behaves like <code>\stex_annotate:nnn</code> $\{\langle\text{property}\rangle\} \{\langle\text{resource}\rangle\} \{\langle\text{content}\rangle\}$.
--------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

3.1.2 Languages

<code>\c_stex_languages_prop</code>
<code>\c_stex_language_abbrevs_prop</code>

Map language abbreviations to their full babel names and vice versa. e.g. `\c_stex_languages_prop{en}` yields `english`, and `\c_stex_language_abbrevs_prop{english}` yields `en`.

3.2 Files, Paths, URIs

<code>\stex_path_from_string:Nn</code>	<code>\stex_path_from_string:Nn</code> $\langle\text{path-variable}\rangle \{\langle\text{string}\rangle\}$
<code>\stex_path_from_string:(NV cn cV)</code>	

turns the $\langle\text{string}\rangle$ into a path by splitting it at `/`-characters and stores the result in $\langle\text{path-variable}\rangle$. Also applies `\stex_path_canonicalize:N`.

<code>\stex_path_to_string:NN</code>
<code>\stex_path_to_string:N</code>

The inverse; turns a path into a string and stores it in the second argument variable, or leaves it in the input stream.

<code>\stex_path_canonicalize:N</code>

Canonicalizes the path provided; in particular, resolves `.` and `..` path segments.

<code>\stex_path_if_absolute_p:N</code> *
<code>\stex_path_if_absolute:NTF</code> *

Checks whether the path provided is *absolute*, i.e. starts with an empty segment

<code>\c_stex_pwd_seq</code>
<code>\c_stex_pwd_str</code>
<code>\c_stex_mainfile_seq</code>

Store the current working directory as path-sequence and string, respectively, and the (heuristically guessed) full path to the main file, based on the PWD and `\jobname`.

<code>\g_stex_currentfile_seq</code>

The file being currently processed (respecting `\input` etc.)

Test 1

```
\ExplSyntaxOn
\def\cpath@print#1{
\stex_path_from_string:Nn \l_tmpb_seq { #1 }
\stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
\str_use:N \l_tmpa_str
}
\ExplSyntaxOff
\begin{center}
\begin{tabular}{|l|l|l|}\hline
path & canonicalized path & expected\\\hline
aaa & \cpath@print{aaa} & aaa \\
../.. / aaa & \cpath@print{../.. / aaa} & ../.. / aaa \\
aaa/bbb & \cpath@print{aaa/bbb} & aaa/bbb \\
aaa/.. & \cpath@print{aaa/..} & \\
../.. / aaa/bbb & \cpath@print{../.. / aaa/bbb} & ../.. / aaa/bbb \\
../aaa / .. / bbb & \cpath@print{../aaa / .. / bbb} & ../ bbb \\
../aaa/bbb & \cpath@print{../aaa/bbb} & ../aaa/bbb \\
aaa/bbb / .. / ddd & \cpath@print{aaa/bbb / .. / ddd} & aaa/ddd \\
aaa/bbb / .. / ddd & \cpath@print{aaa/bbb / .. / ddd} & aaa/bbb/ddd \\
./ & \cpath@print{./} & \\
aaa/bbb / .. / .. & \cpath@print{aaa/bbb / .. / ..} & \\ \hline
\end{tabular}
\end{center}
```

path	canonicalized path	expected
aaa	aaa	aaa
../.. / aaa	../.. / aaa	../.. / aaa
aaa/bbb	aaa/bbb	aaa/bbb
aaa/..		
../.. / aaa/bbb	../.. / aaa/bbb	../.. / aaa/bbb
../aaa / .. / bbb	../ bbb	../ bbb
../aaa/bbb	../aaa/bbb	../aaa/bbb
aaa/bbb / .. / ddd	aaa/ddd	aaa/ddd
aaa/bbb / .. / ddd	aaa/bbb/ddd	aaa/bbb/ddd
./		
aaa/bbb / .. / ..		

3.3 MathHub Archives

`\mathhub`
`\c_stex_mathhub_seq`
`\c_stex_mathhub_str`

We determine the path to the local MathHub folder via one of three means, in order of precedence:

1. The `mathhub` package option, or
2. the `\mathhub`-macro, if it has been defined before the `\usepackage{stex}`-statement, or
3. the `MATHHUB` system variable.

In all three cases, `\c_stex_mathhub_seq` and `\c_stex_mathhub_str` are set accordingly.

`\l_stex_current_repository_prop`

Always points to the *current* MathHub repository (if we currently are in one). Has the fields `id`, `ns` (namespace), `narr` (narrative namespace; currently not in use) and `deps` (dependencies; currently not in use).

`\stex_set_current_repository:n`

Sets the current repository to the one with the provided ID. calls `__stex_mathhub_do_manifest:n`, so works whether this repository's MANIFEST.MF-file has already been read or not.

`\stex_require_repository:n`

Calls `__stex_mathhub_do_manifest:n` iff the corresponding archive property list does not already exist, and adds a corresponding definition to the `.sms`-file.

`\libinput`

`\libinput{<filename>}`

Inputs `<filename>.tex` from the `lib` folders in the current archive and the `meta-inf`-archive of the current archive group (if existent). Throws an error if no file by that name exists in either folder, includes both if both exist.

Test 2

```
\ExplSyntaxOn
\stex_require_repository:n { Foo/Bar }
id:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {id}\ \
narr:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {narr}\ \
ns:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {ns}\ \
deps:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {deps}\ \
\stex_require_repository:n { Bar/Foo }
\ExplSyntaxOff
```

```
id: Foo/Bar
narr:
ns: http://mathhub.info/tests/Foo/Bar
deps:
```

3.4 The Module System

`\l_stex_current_module_prop`

All information of a module is stored as a property list. `\l_stex_current_module_prop` always points to the current module (if existent).

Most importantly, the `content`-field stores all the code to execute on activation; i.e. when this module is being included.

Additionally, it stores:

- The *name* in field `name`,
- the *namespace* in field `ns`,
- this module's *language* in field `lang`,
- if a language module that translates some other modules, the *original* module in field `sig` (for signature),
- the *metatheory* in field `meta`,
- the URIs of all *imported modules* in field `imports`,
- the names of all *declarations* in field `constants`,
- the *file* this module was declared in in field `file`,

`\stex_if_in_module_p: *` Conditional for whether we are currently in a module
`\stex_if_in_module:TF *`

`\stex_if_module_exists_p:n *`
`\stex_if_module_exists:nTF *`

Conditional for whether a module with the provided URI is already known.

`\stex_add_to_current_module:n`

Adds the provided tokens to the `content` field of the current module.

`\stex_add_constant_to_current_module:n`

Adds the declaration with the provided name to the `constants` field of the current module.

`\stex_add_import_to_current_module:n`

Adds the module with the provided full URI to the `imports` field of the current module.

<code>\stex_modules_compute_namespace:nN</code>	<code>\stex_modules_compute_namespace:nN</code> <code>{\langle namespace \rangle} {\langle path \rangle}</code>
-------------------------------------------------	--------------------------------------------------------------------------------------------------------------------

Computes the namespace for file $\langle path \rangle$ in repository with namespace $\langle namespace \rangle$ as follows:

If the file is `.../source/sub/file.tex` and the namespace `http://some.namespace/foo`, then the namespace of is `http://some.namespace/foo/sub/file`.

<code>\stex_modules_current_namespace:</code>

Computes the current namespace

Test 3

```

\ExplSyntaxOn
\stex_modules_current_namespace:
Namespace~1:\\ \l_stex_modules_ns_str \\
Faking~a~repository:\\
\stex_set_current_repository:n{Foo/Bar}
\seq_pop_right:NN \g_stex_currentfile_seq \testtemp
\edef\testtempb{\detokenize{source}}
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtempb }
\edef\testtempb{\detokenize{test}}
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtempb }
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtemp }
\stex_modules_current_namespace:
Namespace~2:\\ \l_stex_modules_ns_str
\ExplSyntaxOff

```

```

Namespace 1:
file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest
Faking a repository:
Namespace 2:
http://mathhub.info/tests/Foo/Bar/test/stextest

```

3.4.1 The module-environment

<code>module</code>	<code>\begin{module}[\langle options \rangle]{\langle name \rangle}</code> Opens a new module with name $\langle name \rangle$. TODO document options.
---------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------

<code>\stex_modules_heading:</code>	Takes care of the module header, if the <code>showmods</code> package option is true. This macro can be overridden for customization.
-------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------

<code>@module</code>	<code>\begin{@module}[\langle options \rangle]{\langle name \rangle}</code> Core functionality of the module-environment without a header.
----------------------	-----------------------------------------------------------------------------------------------------------------------------------------------

Test 4

```
\ExplSyntaxOn
\stex_set_current_repository:n {Foo/Bar}
\seq_pop_right:NN \g_stex_currentfile_seq \l_tmpa_tl
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{tests} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Bar} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{source} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo.tex} }
\begin{@module}{Foo}
Module-path:-
\prop_item:Nn \l_stex_current_module_prop { ns }?
\prop_item:Nn \l_stex_current_module_prop { name }\\
Language:-\prop_item:Nn \l_stex_current_module_prop { lang }\\
Signature:-\prop_item:Nn \l_stex_current_module_prop { sig }\\
Metatheory:-\prop_item:Nn \l_stex_current_module_prop { meta }\\
\end{@module}
\ExplSyntaxOff
```

```
Module path: http://mathhub.info/tests/Foo/Bar?Foo
Language:
Signature:
Metatheory:
```

Test 5

```
\ExplSyntaxOn
\stex_set_current_repository:n {Foo/Bar}
\stex_debug:n{Test:-\stex_path_to_string:N \g_stex_currentfile_seq }
\seq_pop_right:NN \g_stex_currentfile_seq \l_tmpa_tl
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{tests} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Bar} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{source} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo.tex} }
\stex_debug:n{Test:-\stex_path_to_string:N \g_stex_currentfile_seq }
\begin{module}[title=Foo Bar]{Bar}
Module-path:-
\prop_item:Nn \l_stex_current_module_prop { ns }?
\prop_item:Nn \l_stex_current_module_prop { name }\\
Language:-\prop_item:Nn \l_stex_current_module_prop { lang }\\
Signature:-\prop_item:Nn \l_stex_current_module_prop { sig }\\
Metatheory:-\prop_item:Nn \l_stex_current_module_prop { meta }\\
\end{module}
\ExplSyntaxOff
```

```
Module 3.1[Bar] (FooBar)
Module path: http://mathhub.info/tests/Foo/Bar/Foo?Bar
Language:
Signature:
Metatheory:
```

\l_stex_all_modules_seq

Stores full URIs for all modules currently in scope.

\STEXModule

\STEXModule {*<fragment>*}

Attempts to find a module whose URI ends with *<fragment>* in the current scope and passes the full URI on to \stex_invoke_module:n.

`\stex_invoke_module:n`

Invoked by `\STEXModule`. Needs to be followed either by `!\langle macro \rangle` or `?{\langle symbolname \rangle}`. In the first case, it stores the full URI in `\langle macro \rangle`; in the second case, it invokes the symbol `\langle symbolname \rangle` in the selected module.

Test 6

```
\begin{module}{STEXModuleTest1}
\syndeci{foo}
\end{module}
\begin{module}{STEXModuleTest2}
\importmodule{STEXModuleTest1}
\syndeci{foo}
\end{module}
\begin{module}{STEXModuleTest3}
\importmodule{STEXModuleTest2}
\syndeci{foo}
\STEXModule{STEXModuleTest1}!\teststring
\teststring\
\STEXModule{STEXModuleTest2}!\teststring
\teststring\
\STEXModule{STEXModuleTest3}!\teststring
\teststring\
\STEXModule{STEXModuleTest1}?{foo}{\comp{foo1}}\
\STEXModule{STEXModuleTest2}?{foo}{\comp{foo2}}\
\STEXModule{STEXModuleTest3}?{foo}{\comp{foo3}}\
\end{module}
```

Module 3.2[STEXModuleTest1]

Module 3.3[STEXModuleTest2]

Module 3.4[STEXModuleTest3]
file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?STEXModuleTest1
file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?STEXModuleTest2
file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?STEXModuleTest3
foo1
foo2
foo3

3.4.2 SMS Mode

“SMS Mode” is used when loading modules from external tex files. It deactivates any output and ignores all \TeX commands not explicitly allowed via the following lists:

`\g_stex_smsmode_allowedmacros_tl`

Macros that are executed as is; i.e. with the category code scheme used in SMS mode.

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

Importantly, these macros need to call `\stex_smsmode_set_codes:` after reading all arguments. Note, that `\stex_smsmode_set_codes:` takes care of checking whether we are in SMS mode in the first place, so calling this function eagerly is unproblematic.

`\g_stex_smsmode_allowedenvs_seq`

The names of environments that should be allowed in SMS mode. The corresponding `\begin`-statements are treated like the macros in `\g_stex_smsmode_allowedmacros_escape_tl`, so `\stex_smsmode_set_codes:` should be called at the end of the `\begin`-code. Since `\end`-statements take no arguments anyway, those are called with the SMS mode category code scheme active.

`\stex_if_smsmode_p: *`
`\stex_if_smsmode:TF *`

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

Sets the current category code scheme to that of the SMS mode, if SMS mode is currently active and if necessary.

This method should be called at the end of every macro or `\begin` environment code that are allowed in SMS mode.

`\stex_in_smsmode:nn`

`\stex_in_smsmode:nn {<name>} {<code>}`

Executes `<code>` in SMS mode. `<name>` can be arbitrary, but should be distinct, since it allows for nesting `\stex_in_smsmode:nn` without spuriously terminating SMS mode.

Test 7

```
\immediate\openout\testfile=./tests/sometest.tex
\immediate\write\testfile{\detokenize{\this is \a test}^~J}
\immediate\write\testfile{\detokenize{this \is a \test}}
\immediate\closeout\testfile
\ExplSyntaxOn
\stex_in_smsmode:nn { foo } {
  \input{tests/sometest.tex}
}
\ExplSyntaxOff
```

3.4.3 Imports and Inheritance

`\importmodule`

`\importmodule[<archive-ID>]{<module-path>}`

Imports a module by reading it from a file and “activating” it. `\TeX` determines the module and its containing file by passing its arguments on to `\stex_import_module_path:nn`.

Test 8

```
\begin{module}{Foo}
\symdecl[name=foo, args=3]{bar}
\symdecl[ args=bai]{foobar}
Meaning:-\present\bar\
\end{module}
Meaning:-\present\bar\
\begin{module}{Importtest}
\importmodule{Foo}
Meaning:-\present\bar\
\end{module}
\begin{module}{Importtest2}
\importmodule{Importtest}
Meaning:-\present\bar\
\end{module}
```

Module 3.5[Foo]

Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?Foo?foo}<

Meaning: >macro:->\protect \bar <

Module 3.6[Importtest]

Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?Foo?foo}<

Module 3.7[Importtest2]

Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?Foo?foo}<

\usemodule \importmodule[<archive-ID>]{<module-path>}

Like \importmodule, but does not export its contents; i.e. including the current module will not activate the used module

Test 9

```
\begin{module}{UseTest1}
\symdecl{foo}
\end{module}
\begin{module}{UseTest2}
\usemodule{UseTest1}
\symdecl{bar}
Meaning:-\present\foo\
\end{module}
\begin{module}{UseTest3}
\importmodule{UseTest2}
Meaning:-\present\foo\
Meaning:-\present\bar\

All modules: \ExplSyntaxOn
\seq_use:Nn \l_stex_all_modules_seq {,-} \
All-symbols:-
\seq_use:Nn \l_stex_all_symbols_seq {,-}
\ExplSyntaxOff
\end{module}
```

Module 3.8[UseTest1]

Module 3.9[UseTest2]

Meaning: »macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?UseTest1?foo}<

Module 3.10[UseTest3]

Meaning: »undefined<
Meaning: »macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?UseTest2?bar}<
All modules: file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?UseTest3, file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?UseTest2
All symbols: file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?UseTest2?bar

Test 10

Circular dependencies:

```

\begin{module}{CircDep1}
\importmodule[Foo/Bar]{circular1?Circular1}
\importmodule[Bar/Foo]{circular2?Circular2}
\present\fooA\
\present\fooB
\end{module}

```

Circular dependencies:

Module 3.11[CircDep1]

»macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Foo/Bar/circular1?Circular1?fooA}<
»macro:->\stex_invoke_symbol:n {http://mathhub.info/tests/Bar/Foo/circular2?Circular2?fooB}<

`\stex_import_module_uri:nn`

`\stex_import_module_uri:nn {<archive-ID>} {<module-path>}`

Determines the URI of a module by splitting `<module-path>` into `<path>?<name>`. If `<module-path>` does *not* contain a `?`-character, we consider it to be the `<name>`, and `<path>` to be empty.

If `<archive-ID>` is empty, it is automatically set to the ID of the current archive (if one exists).

1. If `<archive-ID>` is empty:

(a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the same folder, containing a module `<name>`.

That module should have the same namespace as the current one.

(b) If `<path>` is not empty, it must point to the relative path of the containing file as well as the namespace.

2. Otherwise:

(a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the top `source` folder of the archive, containing a module `<name>`.

That module should lie directly in the namespace of the archive.

(b) If `<path>` is not empty, it must point to the path of the containing file as well as the namespace, relative to the namespace of the archive.

If a module by that namespace exists, it is returned. Otherwise, we call `\stex_require_module:nn` on the `source` directory of the archive to find the file.

`\stex_import_require_module:nnnn`

`{<ns>} {<archive-ID>} {<path>} {<name>}`

Checks whether a module with URI `<ns>?<name>` already exists. If not, it looks for a plausible file that declares a module with that URI.

Finally, activates that module by executing its `content`-field.

`\g_stex_module_files_prop`

`\g_stex_modules_in_file_seq`

A property list mapping file paths to the lists of all modules declared therein. `\g_stex_modules_in_file_seq` always points to the current file(-stream - `\inputs` are considered the same file).

3.5 Symbols and Terms

`\symdecl` `\symdecl[⟨args⟩]{⟨macroname⟩}`

Declares a new symbol with semantic macro `\macroname`. Optional arguments are:

- **name**: An (OMDOC) name. By default equal to `⟨macroname⟩`.
- **type**: An (ideally semantic) term. Not used by $\text{\S}\text{\TeX}$, but passed on to MMT for semantic services.
- **local**: A boolean (by default false). If set, this declaration will not be added to the module content, i.e. importing the current module will not make this declaration available.
- **args**: Specifies the “signature” of the semantic macro. Can be either an integer $0 \leq n \leq 9$, or a (more precise) sequence of the following characters:
 - i a “normal” argument, e.g. `\symdecl[args=ii]{plus}` allows for `\plus{2}{2}`.
 - a an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. `\symdecl[args=a]{plus}` allows for `\plus{2,2,2}`.
 - b a *variable* argument. Is treated by $\text{\S}\text{\TeX}$ like an i-argument, but an application is turned into an `OMBind` in OMDOC, binding the provided variable in the subsequent arguments of the operator; e.g. `\symdecl[args=bi]{forall}` allows for `\forall{x\in\text{\Nat}}{x\geq 0}`.

`\abbrdef` `\abbrdef[⟨args⟩]{⟨macroname⟩}{⟨term⟩}`

`\abbrdef` behaves like `\symdecl`, but adds the definiens `⟨term⟩` to the symbol. The latter is largely ignored and irrelevant to $\text{\S}\text{\TeX}$, but exported to OMDOC.

`\stex_symdecl_do:n`

Implements the core functionality of `\symdecl`, and is called by `\symdecl`, `\symdef` and `\abbrdef`.

Ultimately stores the symbol `⟨URI⟩` in the property list `\g_stex_symdecl_⟨URI⟩_prop` with fields:

- **name** (string),
- **module** (string),
- **notations** (sequence of strings; initially empty),
- **local** (boolean),
- **type** (token list),
- **args** (string of is, as and bs),
- **arity** (integer string),
- **assocs** (integer string; number of associative arguments),

Test 11

```
\begin{module}{SymdeclTest}
\symdecl[name=foo, args=3]{bar}
\symdecl[name=foobar, args=iab]{bari}
\abbrdef{bardef}{\bar* abc}
\ExplSyntaxOn
Meaning:-\present\bar\
\stex_get_symbol:n { bar }
Result:-\l_stex_get_symbol_uri_str\
Meaning:-\present\bardef\
\ExplSyntaxOff
\end{module}
```

```
Module 3.12[SymdeclTest]
Meaning: »macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?SymdeclTest?foo}<
Result: file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?SymdeclTest?foo
Meaning: »macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/sty/stex-master/stextest?SymdeclTest?bardef}<
```

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

`\stex_get_symbol:n`

Computes the full URI of a symbol from a macro argument, e.g. the macro name, the macro itself, the full URI...

`\STEXsymbol`

Uses `\stex_get_symbol:n` to find the symbol denoted by the first argument and passes the result on to `\stex_invoke_symbol:n`

`\stex_invoke_symbol:n`

Executes a semantic macro. Outside of math mode or if followed by `*`, it continues to `\stex_term_custom:nn`. In math mode, it uses the default or optionally provided notation of the associated symbol.

If followed by `!`, it will invoke the symbol *itself* rather than its application (and continue to `\stex_term_custom:nn`), i.e. it allows to refer to `\plus![addition]` as an operation, rather than `\plus[addition of]{some}{terms}`.

`\notation`

`\notation[⟨args⟩]{⟨symbol⟩}{⟨notations+⟩}`

Introduces a new notation for `⟨symbol⟩`, see `\stex_notation_do:nn`

 $\backslash\text{stex_notation_do:nn}$
 $\backslash\text{stex_notation_do:nn}\langle\text{URI}\rangle\{\langle\text{notations}^+\rangle\}$

Implements the core functionality of $\backslash\text{notation}$, and is called by $\backslash\text{notation}$ and $\backslash\text{symdef}$.

Ultimately stores the notation in the property list $\backslash\text{g_stex_notation_}\langle\text{URI}\rangle\#\langle\text{variant}\rangle\#\langle\text{lang}\rangle_\text{prop}$ with fields:

- symbol (URI string),
- language (string),
- variant (string),
- opprec (integer string),
- argprecs (sequence of integer strings)

Test 12

```
\begin{module}{NotationTest}
\importmodule{Foo}
\notation{foo, prec=500;20x20x20}{bar}{\comp\langle {#1} ^ {#2} _ {#3} \comp\rangle }
\notation{foo, prec=500;20x20x20}{foobar}{\comp\langle #1 \comp\mid [ #2 ] ^ {#3} \comp\rangle }{ {#1}_{\comp
```

Module 3.13[NotationTest]

 $\backslash\text{symdef}$
 $\backslash\text{symdef}[\langle\text{args}\rangle]\{\langle\text{symbol}\rangle\}\{\langle\text{notations}^+\rangle\}$

Combines $\backslash\text{symdecl}$ and $\backslash\text{notation}$ by introducing a new symbol and assigning a new notation for it.

Test 13

```
\begin{module}{SymdefTest}
\symdef[ args=a, prec=50]{ plus }{ #1 }{#1 \comp+ #2}
$\plus{ a,b,c }$
\end{module}
```

Module 3.14[SymdefTest]
 $a+b+c$

 $\backslash\text{stex_term_math_oms:nnnn}$
 $\backslash\text{stex_term_math_oma:nnnn}$
 $\backslash\text{stex_term_math_omb:nnnn}$
 $\langle\text{URI}\rangle\langle\text{fragment}\rangle\langle\text{precedence}\rangle\langle\text{body}\rangle$

Annotates $\langle\text{body}\rangle$ as an OMDOC-term (OMID, OMA or OMBIND, respectively) with head symbol $\langle\text{URI}\rangle$, generated by the specific notation $\langle\text{fragment}\rangle$ with (upwards) operator precedence $\langle\text{precedence}\rangle$. Inserts parentheses according to the current downwards precedence and operator precedence.

<code>\stex_term_math_arg:nnn</code>	<code>\stex_term_arg:nnn⟨int⟩⟨prec⟩⟨body⟩</code>
--------------------------------------	--------------------------------------------------

Annotates $\langle body \rangle$ as the $\langle int \rangle$ th argument of the current OMA or OMBIND, with (downwards) argument precedence $\langle prec \rangle$.

<code>\stex_term_math_assoc_arg:nnnn</code>	<code>\stex_term_arg:nnn⟨int⟩⟨prec⟩⟨notation⟩⟨body⟩</code>
---------------------------------------------	------------------------------------------------------------

Annotates $\langle body \rangle$ as the $\langle int \rangle$ th (associative) *sequence* argument (as comma-separated list of terms) of the current OMA or OMBIND, with (downwards) argument precedence $\langle prec \rangle$ and associative notation $\langle notation \rangle$.

<code>\infprec</code> <code>\neginfprec</code>	Maximal and minimal notation precedences.
---------------------------------------------------	-------------------------------------------

<code>\STEXdobrackets</code>	<code>\STEXdobrackets {⟨body⟩}</code>
------------------------------	---------------------------------------

Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current \TeX brackets (by default (and)), which can be changed temporarily using `\STEXwithbrackets`.

<code>\STEXwithbrackets</code>	<code>\STEXwithbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code>
--------------------------------	--------------------------------------------------------

Temporarily (i.e. within $\langle body \rangle$) sets the brackets used by \TeX for automated bracketing (by default (and)) to $\langle left \rangle$ and $\langle right \rangle$.

Note that $\langle left \rangle$ and $\langle right \rangle$ need to be allowed after `\left` and `\right` in display-mode.

Test 14

```
\begin{module}{MathTest1}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{bar}{\comp\langle {#1} ^ {#2} _ {#3} \comp\rangle }
$\bar{abc}$ and $\bar{[foo] abc}$.
\end{module}
```

Module 3.15[MathTest1]
 $\langle a^b_c \rangle$ and $\langle a^b_c \rangle$.

Test 15

```
\begin{module}{MathTest2}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{foobar}{\comp\langle #1 \comp\mid [ #2 ] ^ {#3} \comp\rangle }{ {#1} _ {com} }
$\foobar a{b,c,d,e,f}g$ and $\foobar[foo] a{b,c}g$ and $\foobar abc$

\symdecl[ args=a ]{ plus }
\symdecl[ args=a ]{ mult }
\notation[prec=50]{plus}{#1}{#1 \comp+ #2}
\notation[prec=100]{mult}{#1}{#1 \comp\cdot #2}
$\plus{a,\mult{b,c}}$ and $\mult{a,\plus{\frac{ab}{\frac{ac}}{}}}$
$\displaystyle \plus{a,\mult{b,c}}$ and
\STEXwithbrackets[]{$\displaystyle
\mult{a,\plus{\frac{ab}{\frac{ac}}{}}}$}
\end{module}
```


4 Implementation

4.1 The \TeX document class

```

1 <*cls>
2 \RequirePackage{expl3, l3keys2e}
3 \ProvidesExplClass{stex}{2021/08/01}{1.9}{bla}
4 \LoadClass[border=1px, varwidth]{standalone}
5 \setlength\textwidth{15cm}
6 %\g@addto@macro{\@parboxrestore}{\setlength\parskip{\baselineskip}}
7
8 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
9 \ProcessOptions
10
11 \RequirePackage{stex}
12 </cls>

```

4.2 Preliminaries

```

13 <*package>
14 \RequirePackage{expl3, l3keys2e}
15 \ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}
16
17 Package options:
18 \keys_define:nn { stex } {
19   debug      .bool_set:N = \c_stex_debug_bool ,
20   showmods   .bool_set:N = \c_stex_showmods_bool ,
21   lang       .clist_set:N = \c_stex_languages_clist ,
22   mathhub    .tl_set_x:N = \mathhub ,
23   sms        .bool_set:N = \c_stex_persist_mode_bool ,
24   image      .bool_set:N = \c_tikzinput_image_bool
25 }
26 \ProcessKeysOptions { stex }

```

\sTeX The \TeX logo:

```

25 \protected\def\stex{%
26   \@ifundefined{texorpdfstring}%
27   {\let\texorpdfstring\@firstoftwo}%
28   }%
29   \texorpdfstring{\raisebox{-.5ex}{S\kern-.5ex\TeX}{sTeX}}\xspace%
30 }
31 \def\sTeX{\stex}

```

(End definition for \sTeX . This function is documented on page 8.)

Messages

```

32 \msg_new:nnn{stex}{debug}{}
33 \msg_new:nnn{stex}{warning/nomathhub}{
34   MATHHUB~system~variable~not~found~and~no~
35   \detokenize{\mathhub}~value~set!
36 }
37 \msg_new:nnn{stex}{error/norepository}{}

```

\stex_debug:n Debug mode

```

38 \cs_new_protected:Nn \stex_debug:n {
39   \bool_if:nT{\c_stex_debug_bool}{

```

```

40 \exp_args:Nnn\msg_set:nnn{stex}{debug}{\\Debug:~#1\\}
41 \msg_term:nn{stex}{debug} % should be \msg_note:nn
42 }
43 }
44
45 \stex_debug:n{Debug~mode~on}

```

(End definition for `\stex_debug:n`. This function is documented on page 8.)

`\c__stex_sms_iow` File variable used for the sms-File

```

46 \iow_new:N \c__stex_sms_iow
47 \AddToHook{begindocument}{
48 \bool_if:NTF \c_stex_persist_mode_bool {
49 \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
50 } {
51 \iow_open:Nn \c__stex_sms_iow {\jobname.sms}
52 }
53 }
54 \AddToHook{enddocument}{
55 \bool_if:NF \c_stex_persist_mode_bool {
56 \iow_close:N \c__stex_sms_iow
57 }
58 }

```

(End definition for `\c__stex_sms_iow`.)

`\stex_addtosms:n`

```

59 \cs_new_protected:Nn \stex_addtosms:n {
60 \bool_if:NF \c_stex_persist_mode_bool {
61 \iow_now:Nn \c__stex_sms_iow { #1 }
62 }
63 }

```

(End definition for `\stex_addtosms:n`. This function is documented on page 8.)

4.2.1 L^AT_EX_ML and S^CA_LT_EX

```

64 \RequirePackage{scalatex}

```

We add the namespace abbreviation `ns:stex="http://kwarc.info/ns/sTeX"` to S^CA_LT_EX:

```

65 \scalatex_add_Namespace:nn{stex}{http://kwarc.info/ns/sTeX}

```

`\if@latexml` Conditionals for L^AT_EX_ML:

```

\latexml_if_p:
\latexml_if:TF
66 \ifcsname if@latexml\endcsname\else
67 \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
68 \fi
69
70 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
71 \if@latexml
72 \prg_return_true:
73 \else:
74 \prg_return_false:
75 \fi:
76 }

```

(End definition for `\if@latexml` and `\latexml_if:TF`. These functions are documented on page 8.)

4.2.2 HTML Annotations

77 $\langle @@=\text{stex_annotate} \rangle$

$\backslash l_stex_annotate_arg_tl$
 $\backslash c_stex_annotate_emptyarg_tl$

Used by annotation macros to ensure that the HTML output to annotate is not empty.

```
78 \tl_new:N \l__stex_annotate_arg_tl
79 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
80   \scalatex_if:TF {
81     \scalatex_direct_HTML:n { \c_ampsand_str lrm; }
82   }{-}
83 }
```

(End definition for $\backslash l_stex_annotate_arg_tl$ and $\backslash c_stex_annotate_emptyarg_tl$.)

$\backslash_stex_annotate_checkempty:n$

```
84 \cs_new_protected:Nn \__stex_annotate_checkempty:n {
85   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
86   \tl_if_empty:NT \l__stex_annotate_arg_tl {
87     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
88   }
89 }
```

(End definition for $\backslash_stex_annotate_checkempty:n$.)

$\backslash stex_annotate:enx$

$\backslash stex_annotate_invisible:n$

$\backslash stex_annotate_invisible:nnn$

We define four macros for introducing attributes in the HTML output. The definitions depend on the “backend” used (L^AT_EX_{ML}, S^CA^LT_EX, p^DF^LA^TE_X).

The p^DF^LA^TE_X-macros largely do nothing; the S^CA^LT_EX-implementations are pretty clear in what they do, the L^AT_EX_{ML}-implementations resort to perl bindings.

```
90 \scalatex_if:TF{
91   \cs_new_protected:Nn \stex_annotate:nnn {
92     \__stex_annotate_checkempty:n { #3 }
93     \scalatex_annotate_HTML:nn {
94       property="stex:#1" ~
95       resource="#2"
96     } {
97       \tl_use:N \l__stex_annotate_arg_tl
98     }
99   }
100   \cs_new_protected:Nn \stex_annotate_invisible:n {
101     \__stex_annotate_checkempty:n { #1 }
102     \scalatex_annotate_HTML:nn {
103       stex:visible="false" ~
104       style:display="none"
105     } {
106       \tl_use:N \l__stex_annotate_arg_tl
107     }
108   }
109   \cs_new_protected:Nn \stex_annotate_invisible:nnn {
110     \__stex_annotate_checkempty:n { #3 }
111     \scalatex_annotate_HTML:nn {
112       property="stex:#1" ~
113       resource="#2" ~
114       stex:visible="false" ~
115       style:display="none"
```

```

116   } {
117     \tl_use:N \l__stex_annotate_arg_tl
118   }
119 }
120 \NewDocumentEnvironment{stex_annotate_env} { m m } {
121   \par
122   \scalatex_annotate_HTML_begin:n {
123     property="stex:#1" ~
124     resource="#2"
125   }
126 }{
127   \scalatex_annotate_HTML_end:
128 }
129 }{
130   \latexml_if:TF {
131     \cs_new_protected:Nn \stex_annotate:nnn {
132       \__stex_annotate_checkempty:n { #3 }
133       \mode_if_math:TF {
134         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
135           \tl_use:N \l__stex_annotate_arg_tl
136         }
137       }{
138         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
139           \tl_use:N \l__stex_annotate_arg_tl
140         }
141       }
142     }
143     \cs_new_protected:Nn \stex_annotate_invisible:n {
144       \__stex_annotate_checkempty:n { #1 }
145       \mode_if_math:TF {
146         \cs:w latexml@invisible@math\cs_end:{
147           \tl_use:N \l__stex_annotate_arg_tl
148         }
149       } {
150         \cs:w latexml@invisible@text\cs_end:{
151           \tl_use:N \l__stex_annotate_arg_tl
152         }
153       }
154     }
155     \cs_new_protected:Nn \stex_annotate_invisible:nnn {
156       \__stex_annotate_checkempty:n { #3 }
157       \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
158         \tl_use:N \l__stex_annotate_arg_tl
159       }
160     }
161     \NewDocumentEnvironment{stex_annotate_env} { m m } {
162       \par\begin{latexml@annotateenv}{#1}{#2}
163     }{
164       \end{latexml@annotateenv}
165     }
166   }{
167     \cs_new_protected:Nn \stex_annotate:nnn {#3}
168     \cs_new_protected:Nn \stex_annotate_invisible:n {}
169     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}

```

```

170 \NewDocumentEnvironment{stex_annotate_env} { m m } {\par}{\}
171 }
172 }

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 8.)

4.2.3 Languages

```

173 <@=stex_language>

```

`\c_stex_languages_prop`

`\c_stex_language_abbrevs_prop`

We store language abbreviations in two (mutually inverse) property lists:

```

174 \prop_const_from_keyval:Nn \c_stex_languages_prop {
175   en = english ,
176   de = ngerman ,
177   ar = arabic ,
178   bg = bulgarian ,
179   ru = russian ,
180   fi = finnish ,
181   ro = romanian ,
182   tr = turkish ,
183   fr = french
184 }
185
186 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
187   english   = en ,
188   ngerman   = de ,
189   arabic    = ar ,
190   bulgarian = bg ,
191   russian   = ru ,
192   finnish   = fi ,
193   romanian  = ro ,
194   turkish   = tr ,
195   french    = fr
196 }
197 % todo: chinese simplified (zhs)
198 %       chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 9.)

we use the `lang-package` option to load the corresponding babel languages:

```

199 \clist_if_empty:NF \c_stex_languages_clist {
200   \clist_clear:N \l_tmpa_clist
201   \clist_map_inline:Nn \c_stex_languages_clist {
202     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
203       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
204     } {
205       \msg_set:nnn{stex}{error/unknownlanguage}{
206         Unknown~language~\l_tmpa_str
207       }
208       \msg_error:nn{stex}{error/unknownlanguage}
209     }
210   }
211   \stex_debug:n {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
212   \RequirePackage[\clist_use:Nn \l_tmpa_clist ,]{babel}
213 }

```

4.3 Files, Paths and URIs

214 `<@=stex_path>`

4.3.1 Generic Path Handling

We treat paths as L^AT_EX3-sequences (of the individual path segments, i.e. separated by a /-character) unix-style; i.e. a path is absolute if the sequence starts with an empty entry.

```

\stex_path_from_string:Nn
\stex_path_from_string:NV
\stex_path_from_string:cn
\stex_path_from_string:cV
215 \cs_new_protected:Nn \stex_path_from_string:Nn {
216   \str_set:Nx \l_tmpa_str { #2 }
217   \str_if_empty:NTF \l_tmpa_str {
218     \seq_clear:N #1
219   }{
220     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
221     \sys_if_platform_windows:T{
222       \seq_clear:N \l_tmpa_tl
223       \seq_map_inline:Nn #1 {
224         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
225         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
226       }
227       \seq_set_eq:NN #1 \l_tmpa_tl
228     }
229     \stex_path_canonicalize:N #1
230   }
231 }
232 \cs_generate_variant:Nn \stex_path_from_string:Nn
233 { NV, cn, cV }

```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 9.)

```

\stex_path_to_string:NN
\stex_path_to_string:N
234 \cs_new_protected:Nn \stex_path_to_string:NN {
235   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
236 }
237
238 \cs_new:Nn \stex_path_to_string:N {
239   \seq_use:Nn #1 /
240 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 9.)

```

\c__stex_path_dot_str . and .., respectively.
\c__stex_path_up_str
241 \str_const:Nn \c__stex_path_dot_str {.}
242 \str_const:Nn \c__stex_path_up_str {...}

```

(End definition for `\c__stex_path_dot_str` and `\c__stex_path_up_str`.)

`\stex_path_canonicalize:N` Canonicalizes the path provided; in particular, resolves . and .. path segments.

```

243 \cs_new_protected:Nn \stex_path_canonicalize:N {
244   \seq_if_empty:NF #1 {
245     \seq_clear:N \l_tmpa_seq
246     \seq_get_left:NN #1 \l_tmpa_tl
247     \str_if_empty:NT \l_tmpa_tl {

```

```

248     \seq_put_right:Nn \l_tmpa_seq {}
249 }
250 \seq_map_inline:Nn #1 {
251   \str_set:Nn \l_tmpa_tl { ##1 }
252   \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
253     \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
254       \seq_if_empty:NTF \l_tmpa_seq {
255         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
256           \c__stex_path_up_str
257         }
258       }{
259         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
260         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
261           \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
262             \c__stex_path_up_str
263           }
264         }{
265           \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
266         }
267       }
268     }{
269       \str_if_empty:NF \l_tmpa_tl {
270         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
271       }
272     }
273   }
274 }
275 \seq_gset_eq:NN #1 \l_tmpa_seq
276 }
277 }

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 9.)

`\stex_path_if_absolute_p:N`
`\stex_path_if_absolute:NTF`

```

278 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
279   \seq_if_empty:NNTF #1 {
280     \prg_return_false:
281   }{
282     \seq_get_left:NN #1 \l_tmpa_tl
283     \str_if_empty:NNTF \l_tmpa_tl {
284       \prg_return_true:
285     }{
286       \prg_return_false:
287     }
288   }
289 }

```

(End definition for `\stex_path_if_absolute:NTF`. This function is documented on page 9.)

4.3.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

290 \str_new:N\l_stex_kpsewhich_return_str
291 \cs_new_protected:Nn \stex_kpsewhich:n {

```

```

292 \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
293 \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
294 \tl_trim_spaces:N \l_stex_kpsewhich_return_str
295 }

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 8.)

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

296 \sys_if_platform_windows:TF{
297   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
298 }{
299   \stex_kpsewhich:n{-var-value~PWD}
300 }
301
302 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
303 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
304 \stex_debug:n {PWD:~\str_use:N\c_stex_pwd_str}

```

(End definition for `\c_stex_pwd_seq` and `\c_stex_pwd_str`. These variables are documented on page 9.)

4.3.3 File Hooks and Tracking

```

305 <@@=stex_files>

```

We introduce hooks for file inputs that keep track of the absolute paths of files used. This will be useful to keep track of modules, their archives, namespaces etc.

Note that the absolute paths are only accurate in `\input`-statements for paths relative to the PWD, so they shouldn't be relied upon in any other setting than for \TeX -purposes.

`\g__stex_files_stack` keeps track of file changes

```

306 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

```

307 \stex_path_from_string:Nn \c_stex_mainfile_seq {
308   \c_stex_pwd_str/\jobname.tex
309 }

```

(End definition for `\c_stex_mainfile_seq`. This variable is documented on page 9.)

`\g_stex_currentfile_seq` Hooks for file inputs that push/pop `\g__stex_files_stack` to update `\c_stex_mainfile_seq`.

```

310 \seq_gclear_new:N\g_stex_currentfile_seq
311 \AddToHook{file/before}{
312   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
313   \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
314     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
315   }{
316     \stex_path_from_string:Nn\g_stex_currentfile_seq{
317       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
318     }
319   }

```



```

320 \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
321 \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
322 }
323 \AddToHook{file/after}{
324   \seq_if_empty:NF\g__stex_files_stack{
325     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
326   }
327   \seq_if_empty:NTF\g__stex_files_stack{
328     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
329   }{
330     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
331     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
332   }
333 }

```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 9.)

4.4 MathHub Repositories

```

334 <@@=stex_mathhub>
\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
335 \str_if_empty:NTF\mathhub{
336   \stex_kpsewhich:n{-var-value~MATHHUB}
337   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
338 }
339 \str_if_empty:NTF\c_stex_mathhub_str{
340   \msg_warning:nn{stex}{warning/nomathhub}
341 }{
342   \stex_debug:n {MathHub:~\str_use:N\c_stex_mathhub_str}
343   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
344 }
345 }{
346   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
347   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
348     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
349       \c_stex_pwd_str/\mathhub
350     }
351   }
352   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
353   \stex_debug:n {MathHub:~\str_use:N\c_stex_mathhub_str}
354 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 10.)

```

\__stex_mathhub_do_manifest:n
355 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
356   \str_set:Nx \l_tmpa_str { #1 }
357   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
358     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
359     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
360     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
361     \__stex_mathhub_find_manifest:N \l_tmpa_seq
362     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {

```

```

363     \msg_set:nnn{stex}{error/norepository}{
364       No~archive~#1~found~in~
365       \stex_path_to_string:N \c_stex_mathhub_str
366     }
367     \msg_error:nn{stex}{error/norepository}
368   } {
369     \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
370   }
371 }
372 }

```

(End definition for __stex_mathhub_do_manifest:n.)

\l_stex_mathhub_manifest_file_seq

```

373 \str_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for \l_stex_mathhub_manifest_file_seq.)

__stex_mathhub_find_manifest:N Attempts to find the MANIFEST.MF in some file path and stores its path in \l__stex_mathhub_manifest_file_seq:

```

374 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
375   \seq_set_eq:NN\l_tmpa_seq #1
376   \bool_set_true:N\l_tmpa_bool
377   \bool_while_do:Nn \l_tmpa_bool {
378     \seq_if_empty:NTF \l_tmpa_seq {
379       \bool_set_false:N\l_tmpa_bool
380     }{
381       \file_if_exist:nTF{
382         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
383       }{
384         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
385         \bool_set_false:N\l_tmpa_bool
386       }{
387         \file_if_exist:nTF{
388           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
389         }{
390           \seq_put_right:Nn\l_tmpa_seq{META-INF}
391           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
392           \bool_set_false:N\l_tmpa_bool
393         }{
394           \file_if_exist:nTF{
395             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
396           }{
397             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
398             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
399             \bool_set_false:N\l_tmpa_bool
400           }{
401             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
402           }
403         }
404       }
405     }
406   }
407   \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
408 }

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

409 `\ior_new:N \c__stex_mathhub_manifest_ior`

(End definition for `\c_stex_mathhub_manifest_ior`.)

`_stex_mathhub_parse_manifest:n` Stores the entries in manifest file in the corresponding property list:

```

410 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
411   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
412   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
413   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
414     \str_set:Nn \l_tmpa_str {#1}
415     \exp_args:NNoo \seq_set_split:Nnn
416       \l_tmpb_seq \c_colon_str \l_tmpa_str
417     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
418       \exp_args:NNe \str_set:Nn \l_tmpb_tl {
419         \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
420       }
421       \exp_args:No \str_case:nnTF \l_tmpa_tl {
422         {id} {
423           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
424             { id } \l_tmpb_tl
425         }
426         {narration-base} {
427           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
428             { narr } \l_tmpb_tl
429         }
430         {source-base} {
431           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
432             { ns } \l_tmpb_tl
433         }
434         {ns} {
435           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
436             { ns } \l_tmpb_tl
437         }
438         {dependencies} {
439           \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
440             { deps } \l_tmpb_tl
441         }
442       }{}{}
443     }{}
444   }
445   \ior_close:N \c__stex_mathhub_manifest_ior
446 }

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

447 \cs_new_protected:Nn \stex_set_current_repository:n {
448   \stex_require_repository:n { #1 }
449   \prop_set_eq:Nc \l_stex_current_repository_prop {
450     c_stex_mathhub_#1_manifest_prop
451   }
452 }

```

(End definition for `\stex_set_current_repository:n`. This function is documented on page 11.)

`\stex_require_repository:n`

```

453 \cs_new_protected:Nn \stex_require_repository:n {
454   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
455     \stex_debug:n{Opening~archive:~#1}
456     \__stex_mathhub_do_manifest:n { #1 }
457     \exp_args:Nx \stex_addtosms:n {
458       \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
459         id   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id },
460         ns   = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns },
461         narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr },
462         deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
463       }
464     }
465   }
466 }

```

(End definition for `\stex_require_repository:n`. This function is documented on page 11.)

`\l_stex_current_repository_prop` Current MathHub repository

```

467 \prop_new:N \l_stex_current_repository_prop
468
469 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
470 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
471   \stex_debug:n{Not~currently~in~a~MathHub~repository}
472 } {
473   \__stex_mathhub_parse_manifest:n { main }
474   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
475   \l_tmpa_str
476   \prop_set_eq:cN { c_stex_mathhub_#1_manifest_prop }
477   \c_stex_mathhub_main_manifest_prop
478   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
479   \stex_debug:n{Current~repository:~
480     \prop_item:Nn \l_stex_current_repository_prop {id}
481   }
482 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 10.)

`\libinput`

```

483 \cs_new_protected:Npn \libinput #1 {
484   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
485     \msg_set:nnn{stex}{error/norepository}{
486       \c_backslash_str libinput~needs~to~be~called~in~an~archive
487     }
488     \msg_error:nn{stex}{error/norepository}
489   }
490   \bool_set_false:N \l_tmpa_bool
491   \tl_clear:N \l_tmpa_tl
492   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
493   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
494   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
495   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {

```

```

496 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
497 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
498 / meta-inf / lib / #1.tex}{
499 \bool_set_true:N \l_tmpa_bool
500 \tl_put_right:Nx \l_tmpa_tl {
501 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
502 / meta-inf / lib / #1.tex}
503 }
504 }{}
505 }
506 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
507 / \l_tmpa_str / lib / #1.tex
508 }{
509 \bool_set_true:N \l_tmpa_bool
510 \tl_put_right:Nx \l_tmpa_tl {
511 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
512 / \l_tmpa_str / lib / #1.tex}
513 }
514 }{}
515 \bool_if:NF \l_tmpa_bool {
516 \msg_set:nnn{stex}{error/nofile}{
517 \c_backslash_str libinput~no~file~#1.tex~found!
518 }
519 \msg_error:nn{stex}{error/nofile}
520 }
521 \l_tmpa_tl
522 }

```

(End definition for `\libinput`. This function is documented on page 11.)

4.5 Module System

```

523 <@@=stex_module>

```

`\l_stex_current_module_prop`

```

524 \prop_new:N \l_stex_current_module_prop

```

(End definition for `\l_stex_current_module_prop`. This variable is documented on page 12.)

`stex_if_in_module_p:`

`stex_if_in_module:TF`

```

525 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
526 \prop_if_empty:NTF \l_stex_current_module_prop
527 \prg_return_false: \prg_return_true:
528 }

```

(End definition for `stex_if_in_module:TF`. This function is documented on page 12.)

`stex_if_module_exists_p:n`

`stex_if_module_exists:nTF`

```

529 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
530 \prop_if_exist:cTF { c_stex_module_#1_prop }
531 \prg_return_true: \prg_return_false:
532 }

```

(End definition for `stex_if_module_exists:nTF`. This function is documented on page 12.)

`\stex_add_to_current_module:n`

```
533 \cs_new_protected:Nn \stex_add_to_current_module:n {
534   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
535   \tl_put_right:Nn \l_tmpa_tl { #1 }
536   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
537 }
```

(End definition for `\stex_add_to_current_module:n`. This function is documented on page 12.)

`\stex_add_constant_to_current_module:n`

```
538 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
539   \str_set:Nx \l_tmpa_str { #1 }
540   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
541   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
542   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
543 }
```

(End definition for `\stex_add_constant_to_current_module:n`. This function is documented on page 12.)

`\stex_add_import_to_current_module:n`

```
544 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
545   \str_set:Nx \l_tmpa_str { #1 }
546   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
547   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
548   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
549 }
```

(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 12.)

`\stex_modules_compute_namespace:nN` stores its return values in:

`\l_stex_modules_ns_str`

```
550 \str_new:N \l_stex_modules_ns_str

551 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
552   \str_set:Nx \l_tmpa_str { #1 }
553   \seq_set_eq:NN \l_tmpa_seq #2
554   % split off file extension
555   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
556   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
557   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
558   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
559
560   \bool_set_true:N \l_tmpa_bool
561   \bool_while_do:Nn \l_tmpa_bool {
562     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
563     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
564       {source} { \bool_set_false:N \l_tmpa_bool }
565     }{}{
566       \seq_if_empty:NT \l_tmpa_seq {
567         \bool_set_false:N \l_tmpa_bool
568       }
569     }
570 }
```

```

571 \seq_if_empty:NTF \l_tmpa_seq {
572   \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
573 }{
574   \str_set:Nx \l_stex_modules_ns_str {
575     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
576   }
577 }
578 }
579 }

```

(End definition for `\stex_modules_compute_namespace:nN` and `\l_stex_modules_ns_str`. These functions are documented on page 13.)

`\stex_modules_current_namespace:`

```

580 \cs_new_protected:Nn \stex_modules_current_namespace: {
581   \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
582     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
583   }{
584     % split off file extension
585     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
586     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
587     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
588     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
589     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
590     \str_set:Nx \l_stex_modules_ns_str {
591       file:\stex_path_to_string:N \l_tmpa_seq
592     }
593   }
594 }

```

(End definition for `\stex_modules_current_namespace:.` This function is documented on page 13.)

4.5.1 The module environment

`\l_stex_all_modules_seq` Stores all available modules

```

595 \seq_new:N \l_stex_all_modules_seq

```

(End definition for `\l_stex_all_modules_seq`. This variable is documented on page 14.)

`\STEXModule`

`\stex_invoke_module:n`

```

596 \NewDocumentCommand \STEXModule { m } {
597   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
598   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
599   \tl_set:Nn \l_tmpa_tl {
600     \msg_set:nnn{stex}{error/unknownmodule}{
601       No~module~#1~found!
602     }
603     \msg_error:nn{stex}{error/unknownmodule}
604   }
605   \seq_map_inline:Nn \l_stex_all_modules_seq {
606     \str_set:Nn \l_tmpb_str { ##1 }
607     \str_if_eq:eeT { \l_tmpa_str } {
608       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
609     } {
610       \seq_map_break:n {

```

```

611         \tl_set:Nn \l_tmpa_tl {
612             \stex_invoke_module:n { ##1 }
613         }
614     }
615 }
616 }
617 \l_tmpa_tl
618 }
619
620 \cs_new_protected:Nn \stex_invoke_module:n {
621     \stex_debug:n{Invoking~module~#1}
622     \peek_charcode_remove:NTF ! {
623         \__stex_module_invoke_uri:nN { #1 }
624     } {
625         \peek_charcode_remove:NTF ? {
626             \__stex_module_invoke_symbol:nn { #1 }
627         } {
628             \msg_set:nnn{stex}{error/syntax}{
629                 Syntax~error:~?~or~!~expected~after~
630                 \c_backslash_str STEXModule{#1}
631             }
632             \msg_error:nn{stex}{error/syntax}
633         }
634     }
635 }
636
637 \cs_new_protected:Nn \__stex_module_invoke_uri:nN {
638     \str_set:Nn #2 { #1 }
639 }
640
641 \cs_new_protected:Nn \__stex_module_invoke_symbol:nn {
642     \stex_invoke_symbol:n{#1?#2}
643 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 14.)

module module arguments:

```

644 \keys_define:nn { stex / module } {
645     title      .tl_set_x:N = \l_stex_module_title_str ,
646     ns         .tl_set_x:N = \l_stex_module_ns_str ,
647     lang       .tl_set_x:N = \l_stex_module_lang_str ,
648     sig        .tl_set_x:N = \l_stex_module_sig_str ,
649     creators   .tl_set_x:N = \l_stex_module_creators_str ,
650     contributors .tl_set_x:N = \l_stex_module_contributors_str ,
651     meta       .tl_set_x:N = \l_stex_module_meta_str
652 }
653
654 % module parameters here? In the body?
655
656 \cs_new_protected:Nn \__stex_module_args:n {
657     \str_clear:N \l_stex_module_title_str
658     \str_clear:N \l_stex_module_ns_str
659     \str_clear:N \l_stex_module_lang_str

```



```

660 \str_clear:N \l_stex_module_sig_str
661 \str_clear:N \l_stex_module_creators_str
662 \str_clear:N \l_stex_module_contributors_str
663 \str_clear:N \l_stex_module_meta_str
664 \keys_set:nn { stex / module } { #1 }
665 \exp_args:NNo \str_set:Nn \l_stex_module_title_str
666   \l_stex_module_title_str
667 \exp_args:NNo \str_set:Nn \l_stex_module_ns_str
668   \l_stex_module_ns_str
669 \exp_args:NNo \str_set:Nn \l_stex_module_lang_str
670   \l_stex_module_lang_str
671 \exp_args:NNo \str_set:Nn \l_stex_module_sig_str
672   \l_stex_module_sig_str
673 \exp_args:NNo \str_set:Nn \l_stex_module_meta_str
674   \l_stex_module_meta_str
675 \exp_args:NNo \str_set:Nn \l_stex_module_creators_str
676   \l_stex_module_creators_str
677 \exp_args:NNo \str_set:Nn \l_stex_module_contributors_str
678   \l_stex_module_contributors_str
679 }

```

`_stex_module_begin_module:` implements `\begin{module}`

```

680 \cs_new_protected:Nn \_stex_module_begin_module: {
681   % Nested module?
682   \stex_if_in_module:TF {
683     % Nested module
684     \prop_get:NnN \l_stex_current_module_prop
685       { ns } \l_stex_module_ns_str
686     \str_set:Nx \l_stex_module_name_str {
687       \prop_item:Nn \l_stex_current_module_prop
688         { name } / \l_stex_module_name_str
689     }
690   }{
691     % not nested:
692     \str_if_empty:NT \l_stex_module_ns_str {
693       \stex_modules_current_namespace:
694       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
695       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
696         / {\l_stex_module_ns_str}
697       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
698       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
699         \str_set:Nx \l_stex_module_ns_str {
700           \stex_path_to_string:N \l_tmpa_seq
701         }
702       }
703     }
704   }
705
706   % language
707   \str_if_empty:NF \l_stex_module_lang_str {
708     \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
709       \l_tmpa_str {
710         \exp_args:Nx \selectlanguage { \l_tmpa_str }
711       } {

```

```

712         \msg_set:nnn{stex}{error/unknownlanguage}{
713             Unknown~language~\l_tmpa_str
714         }
715         \msg_error:nn{stex}{error/unknownlanguage}
716     }
717 }
718
719 % signature
720 \str_if_empty:NF \l_stex_module_sig_str {
721     \str_if_empty:NT \l_stex_module_lang_str {
722         \msg_set:nnn{stex}{error/siglanguage}{
723             Module~\l_stex_module_ns_str?\l_stex_module_name_str~
724             declares~signature~\l_stex_module_sig_str,~but~does~not~
725             declare~its~language
726         }
727         \msg_error:nn{stex}{error/siglanguage}
728     }
729 }
730
731 % metatheory
732 % \str_if_empty:NTF \l_stex_module_meta_str {
733 %
734 % } {
735 %
736 % }
737
738 \str_clear:N \l_tmpa_str
739 \seq_clear:N \l_tmpa_seq
740 \tl_clear:N \l_tmpa_tl
741 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
742     name      = \l_stex_module_name_str ,
743     ns        = \l_stex_module_ns_str ,
744     imports   = \exp_not:o { \l_tmpa_seq } ,
745     constants = \exp_not:o { \l_tmpa_seq } ,
746     content   = \exp_not:o { \l_tmpa_tl } ,
747     file      = \exp_not:o { \g_stex_currentfile_seq } ,
748     lang      = \l_stex_module_lang_str ,
749     sig       = \l_stex_module_sig_str ,
750     meta      = \l_stex_module_meta_str
751 }
752
753 \stex_debug:n{
754     New~module:\\
755     Namespace:~\l_stex_module_ns_str\\
756     Name:~\l_stex_module_name_str\\
757     Language:~\l_stex_module_lang_str\\
758     Signature:~\l_stex_module_sig_str\\
759     Metatheory:~\l_stex_module_meta_str\\
760     File:~\stex_path_to_string:N \g_stex_currentfile_seq
761 }
762
763 \seq_put_right:Nx \l_stex_all_modules_seq {
764     \l_stex_module_ns_str ? \l_stex_module_name_str
765 }

```

```

766 \seq_gput_right:Nx \g_stex_modules_in_file_seq
767 { \l_stex_module_ns_str ? \l_stex_module_name_str }
768
769
770 \stex_if_smsmode:TF {
771   \stex_smsmode_set_codes:
772 } {
773   \begin{stex_annotate_env} {theory} {
774     \l_stex_module_ns_str ? \l_stex_module_name_str
775   }
776
777   \stex_annotate_invisible:nnn{header}{} {
778     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
779     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
780     \str_if_empty:NT \l_stex_module_meta_str {
781       % TODO metatheory
782     }
783   }
784 }
785 }
786 \iffalse \end{stex_annotate_env} \fi % make syntax highlighting work again

```

(End definition for `_stex_module_begin_module:`)

`_stex_module_end_module:` implements `\end{module}`

```

787 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
788 \cs_new_protected:Nn \_stex_module_end_module: {
789   \str_set:Nx \l_tmpa_str {
790     c_stex_module_
791     \prop_item:Nn \l_stex_current_module_prop { ns } ?
792     \prop_item:Nn \l_stex_current_module_prop { name }
793     _prop
794   }
795   \prop_new:c { \l_tmpa_str }
796   \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
797   \stex_debug:n{Closing module~\prop_item:Nn \l_stex_current_module_prop { name }}
798   \stex_if_smsmode:TF {
799     \exp_args:Nx \stex_addtosms:n {
800       \prop_gset_from_keyval:cn {
801         c_stex_module_
802         \prop_item:Nn \l_stex_current_module_prop { ns } ?
803         \prop_item:Nn \l_stex_current_module_prop { name }
804         _prop
805       } {
806         name      = \prop_item:cn { \l_tmpa_str } { name } ,
807         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
808         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
809         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
810         content   = \prop_item:cn { \l_tmpa_str } { content } ,
811         file      = \prop_item:cn { \l_tmpa_str } { file } ,
812         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
813         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
814         meta      = \prop_item:cn { \l_tmpa_str } { meta }
815       }
816     }
817   }

```

```

816     }
817   }{
818     \end{stex_annotate_env}
819   }
820 }

```

(End definition for `_stex_module_end_module:.`)

@module The core environment, with no header

```

821 \NewDocumentEnvironment { @module } { 0{} m } {
822   \str_set:Nx \l_stex_module_name_str { #2 }
823   \par
824   \__stex_module_args:n { #1 }
825   \__stex_module_begin_module:
826 } {
827   \__stex_module_end_module:
828 }

```

\stex_modules_heading: Code for document headers

```

829 \cs_if_exist:NTF \thesection {
830   \newcounter{module}[section]
831 }{
832   \newcounter{module}
833 }
834
835 \bool_if:NT \c_stex_showmods_bool {
836   \latexml_if:F { \RequirePackage{mdframed} }
837 }
838
839 \cs_new_protected:Nn \stex_modules_heading: {
840   \stepcounter{module}
841   \par
842   \bool_if:NT \c_stex_showmods_bool {
843     \noindent{\textbf{Module} ~
844       \cs_if_exist:NT \thesection {\thesection.}
845       \themodule ~ [\l_stex_module_name_str]
846     }
847     % TODO references
848     % \sref@label@id{Module \thesection.\themodule [\module@name]]%
849     \str_if_empty:NTF \l_stex_module_title_str {
850       }{
851         \quad(\l_stex_module_title_str)\hfill
852       }\par
853     }
854   }

```

(End definition for `\stex_modules_heading:.` This function is documented on page 13.)

Finally:

```

855 \NewDocumentEnvironment { module } { 0{} m } {
856   \bool_if:NT \c_stex_showmods_bool {
857     \begin{mdframed}
858   }
859   \begin{@module}[#1]{#2}
860   \stex_modules_heading:

```

```

861 }{
862   \end{@module}
863   \bool_if:NT \c_stex_showmods_bool {
864     \end{mdframed}
865   }
866 }

```

4.5.2 SMS Mode

```

867 <@=stex_smsmode>

```

```

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
868 \tl_new:N \g_stex_smsmode_allowedmacros_tl
869 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
870 \seq_new:N \g_stex_smsmode_allowedenvs_seq
871
872 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
873   \makeatletter
874   \makeatother
875   \ExplSyntaxOn
876   \ExplSyntaxOff
877 }
878
879 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
880   \symdef
881   \abbrdef
882   \importmodule
883   \notation
884   \symdecl
885 }
886
887 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
888   \tl_to_str:n {
889     module,
890     @module
891   }
892 }

```

(End definition for `\g_stex_smsmode_allowedmacros_tl`, `\g_stex_smsmode_allowedmacros_escape_tl`, and `\g_stex_smsmode_allowedenvs_seq`. These variables are documented on page 15.)

```

\stex_if_smsmode_p:
\stex_if_smsmode:TF
893 \bool_new:N \g__stex_smsmode_bool
894 \bool_set_false:N \g__stex_smsmode_bool
895 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
896   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
897 }

```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 16.)

```

\_stex_smsmode_if_catcodes_p: Checks whether the SMS mode category code scheme is active.
\_stex_smsmode_if_catcodes:TF
898 \bool_new:N \g__stex_smsmode_catcode_bool
899 \bool_set_false:N \g__stex_smsmode_catcode_bool
900 \prg_new_conditional:Nnn \_stex_smsmode_if_catcodes: { p, T, F, TF } {
901   \bool_if:NTF \g__stex_smsmode_catcode_bool

```

```

902 \prg_return_true: \prg_return_false:
903 }

```

(End definition for `_stex_smsmode_if_catcodes:TF`.)

`\stex_smsmode_set_codes:`

```

904 \cs_new_protected:Nn \stex_smsmode_set_codes: {
905   \stex_if_smsmode:T {
906     \__stex_smsmode_if_catcodes:F {
907       \bool_gset_true:N \g__stex_smsmode_catcode_bool
908       \exp_after:wN \char_gset_active_eq:NN
909       \c_backslash_str \__stex_smsmode_cs:
910       \tex_global:D \char_set_catcode_active:N \
911       \tex_global:D \char_set_catcode_other:N $
912       \tex_global:D \char_set_catcode_other:N ^
913       \tex_global:D \char_set_catcode_other:N _
914       \tex_global:D \char_set_catcode_other:N &
915       \tex_global:D \char_set_catcode_other:N ##
916     }
917   }
918 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `\stex_smsmode_set_codes:.` This function is documented on page 16.)

`_stex_smsmode_unset_codes:` Sets category code scheme back from the one used in SMS mode.

```

919 \cs_new_protected:Nn \_stex_smsmode_unset_codes: {
920   \__stex_smsmode_if_catcodes:T {
921     \bool_gset_false:N \g__stex_smsmode_catcode_bool
922     \exp_after:wN \tex_global:D \exp_after:wN
923     \char_set_catcode_escape:N \c_backslash_str
924     \tex_global:D \char_set_catcode_math_toggle:N $
925     \tex_global:D \char_set_catcode_math_superscript:N ^
926     \tex_global:D \char_set_catcode_math_subscript:N _
927     \tex_global:D \char_set_catcode_alignment:N &
928     \tex_global:D \char_set_catcode_parameter:N ##
929   }
930 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `_stex_smsmode_unset_codes:.`)

`\stex_in_smsmode:nn`

```

931 \cs_new_protected:Nn \stex_in_smsmode:nn {
932   \vbox_set:Nn \l_tmpa_box {
933     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
934     \bool_gset_true:N \g__stex_smsmode_bool
935     \stex_smsmode_set_codes:
936     #2
937     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
938     \stex_if_smsmode:F {
939       \_stex_smsmode_unset_codes:
940     }
941   }
942   \box_clear:N \l_tmpa_box
943 }

```

(End definition for `\stex_in_smsmode:n`. This function is documented on page 16.)

`__stex_smsmode_cs:` is executed on encountering `\` in `smsmode`. It checks whether the corresponding command is allowed and executes or ignores it accordingly:

```

944 \cs_new_protected:Nn \__stex_smsmode_cs: {
945   \str_clear:N \l_tmpa_str
946   \peek_analysis_map_inline:n {
947     % #1: token (one expansion)
948     % #2: charcode
949     % #3 catcode
950     \token_if_eq_charcode:NNTF ##3 B {
951       % token is a letter
952       \exp_args:NNo \str_put_right:Nn \l_tmpa_str { ##1 }
953     } {
954       \str_if_empty:NTF \l_tmpa_str {
955         % we don't allow (or need) single non-letter CSs
956         % for now
957         \peek_analysis_map_break:
958       }{
959         \str_if_eq:onTF \l_tmpa_str { begin } {
960           \peek_analysis_map_break:n {
961             \exp_after:wN \__stex_smsmode_checkbegin:n ##1
962           }
963         } {
964           \str_if_eq:onTF \l_tmpa_str { end } {
965             \peek_analysis_map_break:n {
966               \exp_after:wN \__stex_smsmode_checkend:n ##1
967             }
968           } {
969             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
970             \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
971               \g_stex_smsmode_allowedmacros_tl
972               { \use:c{\l_tmpa_str} } {
973               \stex_debug:n{Executing~1:~\l_tmpa_str}
974               \peek_analysis_map_break:n {
975                 \exp_after:wN \l_tmpa_tl ##1
976               }
977             } {
978               \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
979               \g_stex_smsmode_allowedmacros_escape_tl
980               { \use:c{\l_tmpa_str} } {
981               \stex_debug:n{Executing~2:~\l_tmpa_str}
982               % TODO \__stex_smsmode_rescan_cs:
983               \exp_after:wN \exp_after:wN \exp_after:wN
984               \token_if_eq_charcode:NNTF \exp_after:wN \c_backslash_str ##1 {
985               \peek_analysis_map_break:n {
986                 \__stex_smsmode_unset_codes:
987                 \__stex_smsmode_rescan_cs:
988               }
989             } {
990               \peek_analysis_map_break:n {
991                 \__stex_smsmode_unset_codes:
992                 \exp_after:wN \l_tmpa_tl ##1
993               }

```

```

994 %           }
995           } {
996           \peek_analysis_map_break:n { ##1 }
997         }
998       }
999     }
1000   }
1001 }
1002 }
1003 }
1004 }

```

(End definition for `_stex_smsmode_cs:.`)

`_stex_smsmode_rescan_cs:` If the last token gobbled by `\stex_smsmode_cs:` happened to be a `\`, we need to rescan the cs name and reinsert it into the input stream:

```

1005 \cs_new_protected:Nn \_stex_smsmode_rescan_cs: {
1006   \str_clear:N \l_tmpb_str
1007   \peek_analysis_map_inline:n {
1008     \token_if_eq_charcode:NNTF ##3 B {
1009       % token is a letter
1010       \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1011     } {
1012       \peek_analysis_map_break:n {
1013         \exp_after:wN \use:c \exp_after:wN {
1014           \exp_after:wN \l_tmpa_str\exp_after:wN
1015         } \use:c { \l_tmpb_str \exp_after:wN } ##1
1016       }
1017     }
1018   }
1019 }

```

(End definition for `_stex_smsmode_rescan_cs:.`)

`_stex_smsmode_checkbegin:n` called on `\begin`; checks whether the environment being opened is allowed in SMS mode.

```

1020 \cs_new_protected:Nn \_stex_smsmode_checkbegin:n {
1021   \str_set:Nn \l_tmpa_str { #1 }
1022   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1023     \_stex_smsmode_unset_codes:
1024     \begin{#1}
1025   }
1026 }

```

(End definition for `_stex_smsmode_checkbegin:n.`)

`_stex_smsmode_checkend:n` called on `\end`; checks whether the environment being opened is allowed in SMS mode.

```

1027 \cs_new_protected:Nn \_stex_smsmode_checkend:n {
1028   \str_set:Nn \l_tmpa_str { #1 }
1029   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1030     \end{#1}
1031   }
1032 }

```

(End definition for `_stex_smsmode_checkend:n.`)

4.5.3 Inheritance

1033 <@@=stex_importmodule>

\stex_import_module_uri:nn

```

1034 \cs_new_protected:Nn \stex_import_module_uri:nn {
1035   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1036   \str_set:Nx \l__stex_importmodule_path_str { #2 }
1037   \str_if_empty:NT \l__stex_importmodule_archive_str {
1038     \prop_if_empty:NF \l_stex_current_repository_prop {
1039       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
1040     }
1041   }
1042
1043   \exp_args:NNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1044   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1045   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1046
1047   \str_if_empty:NTF \l__stex_importmodule_archive_str {
1048     \stex_modules_current_namespace:
1049     \str_if_empty:NF \l__stex_importmodule_path_str {
1050       \str_set:Nx \l_stex_module_ns_str {
1051         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1052       }
1053     }
1054   }{
1055     \stex_require_repository:n \l__stex_importmodule_archive_str
1056     \prop_get:cnN { c_stex_mathhub\l__stex_importmodule_archive_str _manifest_prop } { ns }
1057     \l_stex_module_ns_str
1058     \str_if_empty:NF \l__stex_importmodule_path_str {
1059       \str_set:Nx \l_stex_module_ns_str {
1060         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1061       }
1062     }
1063   }
1064 }
```

(End definition for \stex_import_module_uri:nn. This function is documented on page 19.)

\l_stex_importmodule_name_str
\l_stex_importmodule_archive_str
\l_stex_importmodule_path_str
\l_stex_importmodule_file_str

Store the return values of \stex_import_module_uri:nn.

```

1065 \str_new:N \l__stex_importmodule_name_str
1066 \str_new:N \l__stex_importmodule_archive_str
1067 \str_new:N \l__stex_importmodule_path_str
1068 \str_new:N \g__stex_importmodule_file_str
```

(End definition for \l__stex_importmodule_name_str and others.)

\stex_import_require_module:nnnn

{<ns>} {<archive-ID>} {<path>} {<name>}

```

1069 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1070   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1071     % \stex_debug:n{Arguments: #1, #2, #3, #4}
1072
1073     % archive
1074     \str_set:Nx \l_tmpa_str { #2 }
1075     \str_if_empty:NTF \l_tmpa_str {
```

```

1076 \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1077 } {
1078 \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1079 \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1080 \seq_put_right:Nn \l_tmpa_seq { source }
1081 }
1082
1083 % path
1084 \str_set:Nx \l_tmpb_str { #3 }
1085 \str_if_empty:NTF \l_tmpb_str {
1086 \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1087
1088 \cs_if_exist:NTF \language {
1089 \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1090 { \language } \l_tmpb_str {
1091 \msg_set:nnn{stex}{error/unknownlanguage}{
1092 Unknown-language-\language
1093 }
1094 \msg_error:nn{stex}{error/unknownlanguage}
1095 }
1096 } {
1097 \str_clear:N \l_tmpb_str
1098 }
1099
1100 \stex_debug:n{Checking-\l_tmpa_str.\l_tmpb_str.tex}
1101 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1102 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1103 }{
1104 \stex_debug:n{Checking-\l_tmpa_str.tex}
1105 \IfFileExists{ \l_tmpa_str.tex }{
1106 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1107 }{
1108 % try english as default
1109 \stex_debug:n{Checking-\l_tmpa_str.en.tex}
1110 \IfFileExists{ \l_tmpa_str.en.tex }{
1111 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1112 }{
1113 \msg_set:nnn{stex}{error/modulemissing}{
1114 No-file-for-module-#1?#4-found
1115 }
1116 \msg_error:nn{stex}{error/modulemissing}
1117 }
1118 }
1119 }
1120
1121 } {
1122 \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1123 \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1124
1125 \cs_if_exist:NTF \language {
1126 \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1127 { \language } \l_tmpb_str {
1128 \msg_set:nnn{stex}{error/unknownlanguage}{
1129 Unknown-language-\language

```

```

1130     }
1131     \msg_error:nn{stex}{error/unknownlanguage}
1132   }
1133 } {
1134   \str_clear:N \l_tmpb_str
1135 }
1136
1137 \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1138
1139 \stex_debug:n{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1140 \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1141   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1142 }{
1143   \stex_debug:n{Checking~\l_tmpa_str/#4.tex}
1144   \IfFileExists{ \l_tmpa_str/#4.tex }{
1145     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1146   }{
1147     % try english as default
1148     \stex_debug:n{Checking~\l_tmpa_str/#4.en.tex}
1149     \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1150       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1151     }{
1152       \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1153       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1154         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1155       }{
1156         \stex_debug:n{Checking~\l_tmpa_str.tex}
1157         \IfFileExists{ \l_tmpa_str.tex }{
1158           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1159         }{
1160           % try english as default
1161           \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1162           \IfFileExists{ \l_tmpa_str.en.tex }{
1163             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1164           }{
1165             \msg_set:nnn{stex}{error/modulemissing}{
1166               No~file~for~module~#1?#4~found
1167             }
1168             \msg_error:nn{stex}{error/modulemissing}
1169           }
1170         }
1171       }
1172     }
1173   }
1174 }
1175 }
1176
1177 \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1178 \seq_clear:N \g_stex_modules_in_file_seq
1179 % \exp_args:Nnx \use:nn {
1180   \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1181     \prop_clear:N \l_stex_current_module_prop
1182     \str_set:Nx \l_tmpb_str { #2 }
1183     \str_if_empty:NF \l_tmpb_str {

```

```

1184         \stex_set_current_repository:n { #2 }
1185     }
1186     \stex_debug:n{Loading~\g__stex_importmodule_file_str}
1187     \input { \g__stex_importmodule_file_str }
1188 }
1189 % }{
1190
1191 % }
1192 \prop_gput:Noo \g_stex_module_files_prop
1193 \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1194 \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1195
1196 \stex_if_module_exists:nF { #1 ? #4 } {
1197     \msg_set:nnn{stex}{error/modulemissing}{
1198         Module~#1?#4~not~found~in~file~\g__stex_importmodule_file_str
1199     }
1200     \msg_error:nn{stex}{error/modulemissing}
1201 }
1202 }
1203 % activate
1204 \stex_debug:n{Activating~module~#1?#4}
1205 \seq_put_right:Nx \l_stex_all_modules_seq {
1206     #1 ? #4
1207 }
1208 \prop_item:cn { c_stex_module_#1?#4_prop } { content }
1209 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 19.)

`\importmodule`

```

1210 \NewDocumentCommand \importmodule { 0{} m } {
1211     \stex_import_module_uri:nn { #1 } { #2 }
1212     \stex_debug:n{Importing~module:~
1213         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1214     }
1215     \stex_if_smsmode:F {
1216         \stex_import_require_module:nnnn
1217         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1218         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1219         \stex_annotate_invisible:nnn
1220         {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1221     }
1222     \exp_args:Nx \stex_add_to_current_module:n {
1223         \stex_import_require_module:nnnn
1224         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1225         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1226     }
1227     \exp_args:Nx \stex_add_import_to_current_module:n {
1228         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1229     }
1230     \stex_smsmode_set_codes:
1231 }

```

(End definition for `\importmodule`. This function is documented on page 16.)

`\usemodule`

```
1232 \NewDocumentCommand \usemodule { 0{} m } {  
1233   \stex_if_smsmode:F {  
1234     \stex_import_module_uri:nn { #1 } { #2 }  
1235     \stex_import_require_module:nnnn  
1236     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }  
1237     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }  
1238     \stex_annotate_invisible:nnn  
1239     {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}  
1240   }  
1241   \stex_smsmode_set_codes:  
1242 }
```

(End definition for `\usemodule`. This function is documented on page 17.)

`\g_stex_modules_in_file_seq`
`\g_stex_module_files_prop`

```
1243 \seq_new:N \g_stex_modules_in_file_seq  
1244 \prop_new:N \g_stex_module_files_prop
```

(End definition for `\g_stex_modules_in_file_seq` and `\g_stex_module_files_prop`. These variables are documented on page 19.)

4.6 Symbol Declarations

```
1245 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1246 \prop_new:N \l_stex_all_symbols_seq
```

(End definition for `\l_stex_all_symbols_seq`. This variable is documented on page 21.)

`\STEXsymbol`

```
1247 \NewDocumentCommand \STEXsymbol { m } {  
1248   \stex_get_symbol:n { #1 }  
1249   \exp_args:No  
1250   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }  
1251 }
```

(End definition for `\STEXsymbol`. This function is documented on page 21.)

symdecl arguments:

```
1252 \keys_define:nn { stex / symdecl } {  
1253   name      .tl_set:x:N = \l_stex_symdecl_name_str ,  
1254   local     .bool_set:N = \l_stex_symdecl_local_bool ,  
1255   args      .tl_set:x:N = \l_stex_symdecl_args_str ,  
1256   type      .tl_set:N   = \l_stex_symdecl_type_tl ,  
1257   align     .tl_set:N   = \l_stex_symdecl_align_str , % TODO(?)  
1258   gfc       .tl_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)  
1259   specializes .tl_set:N = \l_stex_symdecl_specializes_str , % TODO(?)  
1260 }  
1261  
1262 \bool_new:N \l_stex_symdecl_make_macro_bool  
1263  
1264 \cs_new_protected:Nn \__stex_symdecl_args:n {  
1265   \str_clear:N \l_stex_symdecl_name_str  
1266   \str_clear:N \l_stex_symdecl_args_str
```

```

1267 \bool_set_false:N \l_stex_symdecl_local_bool
1268 \tl_clear:N \l_stex_symdecl_type_tl
1269
1270 \keys_set:nn { stex /symdecl } { #1 }
1271
1272 \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
1273   \l_stex_symdecl_name_str
1274 \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
1275   \l_stex_symdecl_args_str
1276 }

```

\symdecl Parses the optional arguments and passes them on to `\stex_symdecl_do:` (so that `\symdef` and `\abbrdef` can do the same)

```

1277 \cs_new_protected:Npn \symdecl {
1278   \peek_charcode_remove:NTF * {
1279     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1280     \__stex_symdecl_:
1281   } {
1282     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1283     \__stex_symdecl_:
1284   }
1285 }
1286
1287 \NewDocumentCommand \__stex_symdecl_: { 0{} m } {
1288   \__stex_symdecl_args:n { #1 }
1289   \tl_clear:N \l_stex_symdecl_definiens_tl
1290   \stex_symdecl_do:n { #2 }
1291 }

```

(End definition for `\symdecl`. This function is documented on page 20.)

\abbrdef

```

1292 \NewDocumentCommand \abbrdef { 0{} m m } {
1293   \__stex_symdecl_args:n { #1 }
1294   \tl_set:Nn \l_stex_symdecl_definiens_tl { #3 }
1295   \bool_set_true:N \l_stex_symdecl_make_macro_bool
1296   \stex_symdecl_do:n { #2 }
1297 }

```

(End definition for `\abbrdef`. This function is documented on page 20.)

\stex_symdecl_do:n

```

1298 \cs_new_protected:Nn \stex_symdecl_do:n {
1299   \stex_if_in_module:F {
1300     % TODO throw error? some default namespace?
1301   }
1302
1303   \str_if_empty:NT \l_stex_symdecl_name_str {
1304     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1305   }
1306
1307   \prop_if_exist:cT { g_stex_symdecl_
1308     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1309     \prop_item:Nn \l_stex_current_module_prop {name} ?

```

```

1310     \l_stex_symdecl_name_str
1311     _prop
1312   }{
1313     % TODO throw error (beware of circular dependencies)
1314   }
1315
1316   \prop_clear:N \l_tmpa_prop
1317   \prop_put:Nnx \l_tmpa_prop { module } {
1318     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1319     \prop_item:Nn \l_stex_current_module_prop {name}
1320   }
1321   \seq_clear:N \l_tmpa_seq
1322   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1323   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1324   \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1325   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1326
1327   \exp_args:No \stex_add_constant_to_current_module:n {
1328     \l_stex_symdecl_name_str
1329   }
1330
1331   % arity/args
1332   \int_zero:N \l_tmpb_int
1333
1334   \bool_set_true:N \l_tmpa_bool
1335   \str_map_inline:Nn \l_stex_symdecl_args_str {
1336     \token_case_meaning:NnF ##1 {
1337       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1338       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1339       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1340       {\tl_to_str:n a} {
1341         \bool_set_false:N \l_tmpa_bool
1342         \int_incr:N \l_tmpb_int
1343       }
1344     }{
1345       \msg_set:nnn{stex}{error/wrongargs}{
1346         args~value~in~symbol~declaration~for~
1347         \prop_item:Nn \l_stex_current_module_prop {ns} ?
1348         \prop_item:Nn \l_stex_current_module_prop {name} ?
1349         \l_stex_symdecl_name_str ~
1350         needs~to~be~
1351         i,~a~or~b,~but~##1~given
1352       }
1353       \msg_error:nn{stex}{error/wrongargs}
1354     }
1355   }
1356   \bool_if:NTF \l_tmpa_bool {
1357     % possibly numeric
1358     \str_if_empty:NTF \l_stex_symdecl_args_str {
1359       \prop_put:Nnn \l_tmpa_prop { args } {}
1360       \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1361     }{
1362       \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1363       \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }

```

```

1364     \str_clear:N \l_tmpa_str
1365     \int_step_inline:nn \l_tmpa_int {
1366         \str_put_right:Nn \l_tmpa_str i
1367     }
1368     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1369 }
1370 } {
1371     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1372     \prop_put:Nnx \l_tmpa_prop { arity }
1373     { \str_count:N \l_stex_symdecl_args_str }
1374 }
1375 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1376
1377
1378 % semantic macro
1379
1380 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1381     \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1382         \prop_item:Nn \l_tmpa_prop { module } ?
1383         \prop_item:Nn \l_tmpa_prop { name }
1384     } }
1385
1386     \bool_if:NF \l_stex_symdecl_local_bool {
1387         \exp_args:Nx \stex_add_to_current_module:n {
1388             \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1389                 \prop_item:Nn \l_tmpa_prop { module } ?
1390                 \prop_item:Nn \l_tmpa_prop { name }
1391             } }
1392         }
1393     }
1394 }
1395
1396 % add to all symbols
1397
1398 \bool_if:NF \l_stex_symdecl_local_bool {
1399     \exp_args:Nx \stex_add_to_current_module:n {
1400         \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1401             \prop_item:Nn \l_tmpa_prop { module } ?
1402             \prop_item:Nn \l_tmpa_prop { name }
1403         }
1404     }
1405 }
1406
1407 \stex_debug:n{New~symbol:~
1408     \prop_item:Nn \l_tmpa_prop { module } ?
1409     \prop_item:Nn \l_tmpa_prop { name }^^J
1410     Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1411     Args:~\prop_item:Nn \l_tmpa_prop { args }
1412 }
1413
1414 \prop_gset_eq:cN {
1415     g_stex_symdecl_
1416     \prop_item:Nn \l_tmpa_prop { module } ?
1417     \prop_item:Nn \l_tmpa_prop { name }

```



```

1418   _prop
1419 } \l_tmpa_prop
1420
1421 \stex_if_smsmode:TF {
1422   \bool_if:NF \l_stex_symdecl_local_bool {
1423     \exp_args:Nx \stex_addtosms:n {
1424       \prop_gset_from_keyval:cn {
1425         g_stex_symdecl_
1426         \prop_item:Nn \l_tmpa_prop { module } ?
1427         \prop_item:Nn \l_tmpa_prop { name }
1428         _prop
1429       } {
1430         name      = \prop_item:Nn \l_tmpa_prop { name }
1431         module    = \prop_item:Nn \l_tmpa_prop { module }
1432         notations = \prop_item:Nn \l_tmpa_prop { notations }
1433         local     = \prop_item:Nn \l_tmpa_prop { local }
1434         type      = \prop_item:Nn \l_tmpa_prop { type }
1435         args      = \prop_item:Nn \l_tmpa_prop { args }
1436         arity     = \prop_item:Nn \l_tmpa_prop { arity }
1437         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1438       }
1439       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1440         \prop_item:Nn \l_tmpa_prop { module } ?
1441         \prop_item:Nn \l_tmpa_prop { name }
1442       }
1443     }
1444   }
1445   \stex_smsmode_set_codes:
1446 }{
1447   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1448     \prop_item:Nn \l_tmpa_prop { module } ?
1449     \prop_item:Nn \l_tmpa_prop { name }
1450   }
1451   \stex_annotate_invisible:nnn {symdecl} {
1452     \prop_item:Nn \l_tmpa_prop { module } ?
1453     \prop_item:Nn \l_tmpa_prop { name }
1454   } {
1455     \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
1456     \stex_annotate_invisible:nnn{args}{}{
1457       \prop_item:Nn \l_tmpa_prop { args }
1458     }
1459     \stex_annotate_invisible:nnn{macroname}{}{#1}
1460     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1461       \stex_annotate_invisible:nnn{definiens}{}
1462       {\l_stex_symdecl_definiens_tl$}
1463     }
1464   }
1465 }
1466 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 20.)

`\stex_get_symbol:n`

```

1467 \str_new:N \l_stex_get_symbol_uri_str

```

```

1468
1469 \cs_new_protected:Nn \stex_get_symbol:n {
1470   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1471     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1472   }{
1473     % argument is a string
1474     % is it a command name?
1475     \cs_if_exist:cTF { #1 }{
1476       \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1477     }{
1478       % argument is not a command name
1479       \prop_get:NnN \l_stex_current_module_prop
1480         { constants } \l_tmpa_seq
1481       \seq_if_in:NnTF \l_tmpa_seq { #1 } {
1482         \str_set:Nx \l_stex_get_symbol_uri_str {
1483           \prop_item:Nn \l_stex_current_module_prop { ns } ?
1484           \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1485         }
1486       } {
1487         \tl_set:Nn \l_tmpa_tl {
1488           \msg_set:nnn{stex}{error/unknownsymbol}{
1489             No~symbol~#1~found!
1490           }
1491           \msg_error:nn{stex}{error/unknownsymbol}
1492         }
1493         \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1494         \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1495         \seq_map_inline:Nn \l_stex_all_symbols_seq {
1496           \str_set:Nn \l_tmpb_str { ##1 }
1497           \str_if_eq:eeT { \l_tmpa_str } {
1498             \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1499           } {
1500             \seq_map_break:n {
1501               \tl_set:Nn \l_tmpa_tl {
1502                 \str_set:Nn \l_stex_get_symbol_uri_str {
1503                   ##1
1504                 }
1505             }
1506           }
1507         }
1508       }
1509       \l_tmpa_tl
1510     }
1511     % \l_stex_all_symbols_seq
1512   }
1513 }
1514 }
1515
1516 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1517   \tl_set:Nx \l_tmpa_tl { #1 }
1518   \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
1519     \stex_invoke_symbol:n {
1520       \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1521         { \tl_tail:N \l_tmpa_tl }

```

```

1522 \tl_if_single:NTF \l_tmpa_tl {
1523   \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1524     \exp_after:wN \str_set:Nn \exp_after:wN
1525       \l_stex_get_symbol_uri_str \l_tmpa_tl
1526   }{
1527     % TODO
1528     % tail is not a single group
1529   }
1530 }{
1531   % TODO
1532   % tail is not a single group
1533 }
1534 }{
1535   % TODO
1536   % head is not \stex_invoke_symbol:n
1537 }
1538 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 21.)

4.7 Notations

```

1539 <@=stex_notation>
      notation arguments:
1540 \keys_define:nn { stex / notation } {
1541   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
1542   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
1543   prec      .tl_set_x:N = \l__stex_notation_prec_str ,
1544   unknown   .code:n      = \str_set:Nx
1545             \l__stex_notation_variant_str \l_keys_key_str
1546 }
1547
1548 \cs_new_protected:Nn \__stex_notation_args:n {
1549   \str_clear:N \l__stex_notation_lang_str
1550   \str_clear:N \l__stex_notation_variant_str
1551   \str_clear:N \l__stex_notation_prec_str
1552
1553   \keys_set:nn { stex / notation } { #1 }
1554
1555   \str_set:Nx \l__stex_notation_lang_str \l__stex_notation_lang_str
1556   \str_set:Nx \l__stex_notation_variant_str \l__stex_notation_variant_str
1557   \str_set:Nx \l__stex_notation_prec_str \l__stex_notation_prec_str
1558 }

```

\notation

```

1559 \NewDocumentCommand \notation { 0{} m } {
1560   \__stex_notation_args:n { #1 }
1561   \tl_clear:N \l_stex_symdecl_definiens_tl
1562   \stex_get_symbol:n { #2 }
1563   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1564 }

```

(End definition for `\notation`. This function is documented on page 21.)

`\stex_notation_do:nn`

```
1565 \cs_new_protected:Nn \stex_notation_do:nn {
1566   \prop_set_eq:Nc \l_tmpa_prop {
1567     g_stex_symdecl_ #1 _prop
1568   }
1569
1570   \prop_clear:N \l_tmpb_prop
1571   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1572   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1573   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1574
1575   % precedences
1576   \seq_clear:N \l_tmpb_seq
1577   \exp_args:NNno
1578   \str_if_empty:NTF \l__stex_notation_prec_str {
1579     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1580     \int_compare:nNnTF \l_tmpa_str = 0 {
1581       \exp_args:NNnx
1582       \prop_put:Nnn \l_tmpb_prop { opprec }
1583       { \int_use:N \infpref }
1584     }{
1585       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1586     }
1587   } {
1588     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
1589     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1590       \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
1591       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
1592         \exp_args:NNno \exp_args:NNno \seq_set_split:Nnn
1593         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
1594         \seq_map_inline:Nn \l_tmpa_seq {
1595           \seq_put_right:Nn \l_tmpb_seq { ##1 }
1596         }
1597       }
1598       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1599     }{
1600       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1601       \int_compare:nNnTF \l_tmpa_str = 0 {
1602         \exp_args:NNnx
1603         \prop_put:Nnn \l_tmpb_prop { opprec }
1604         { \int_use:N \infpref }
1605       }{
1606         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1607       }
1608     }
1609   }
1610
1611   \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1612   \int_step_inline:nn { \l_tmpa_str } {
1613     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
1614       \exp_args:NNx
1615       \seq_put_right:Nn \l_tmpb_seq {
1616         \prop_item:Nn \l_tmpb_prop { opprec }
1617       }
1618     }
1619   }
```

```

1618     }
1619 }
1620
1621 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
1622 \tl_clear:N \l_tmpa_tl
1623
1624 \int_compare:nNnTF \l_tmpa_str = 0 {
1625   \cs_set:Npx \l__stex_notation_macrocode_cs {
1626     \stex_term_math_oms:nnnn { #1 }
1627     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1628     { \prop_item:Nn \l_tmpb_prop { opprec } }
1629     { #2 }
1630   }
1631   \__stex_notation_final:
1632 }{
1633   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1634   \str_if_in:NnTF \l_tmpb_str b {
1635     \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1636     \cs_set:Npx \l_tmpa_str {
1637       \stex_term_math_omb:nnnn { #1 }
1638       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1639       { \prop_item:Nn \l_tmpb_prop { opprec } }
1640       { #2 }
1641     }
1642   }{
1643     \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1644     \cs_set:Npx \l_tmpa_str {
1645       \stex_term_math_oma:nnnn { #1 }
1646       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1647       { \prop_item:Nn \l_tmpb_prop { opprec } }
1648       { #2 }
1649     }
1650   }
1651
1652   \int_zero:N \l_tmpa_int
1653   \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1654   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1655   \__stex_notation_arguments:
1656 }
1657 }

```

(End definition for `\stex_notation_do:nn`. This function is documented on page 22.)

`__stex_notation_arguments:` Takes care of annotating the arguments in a notation macro

```

1658 \cs_new_protected:Nn \__stex_notation_arguments: {
1659   \int_incr:N \l_tmpa_int
1660   \str_if_empty:NnTF \l_tmpa_str {
1661     \__stex_notation_final:
1662   }{
1663     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1664     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1665     \str_if_eq:NnTF \l_tmpb_str a {
1666       \__stex_notation_argument_assoc:n
1667     }{

```

```

1668 \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1669 \tl_put_right:Nx \l_tmpa_tl {
1670   { \stex_term_math_arg:nnn
1671     { \int_use:N \l_tmpa_int }
1672     { \l_tmpb_str }
1673     { ####\int_use:N \l_tmpa_int }
1674   }
1675 }
1676 \__stex_notation_arguments:
1677 }
1678 }
1679 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

1680 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
1681   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1682   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
1683   \tl_put_right:Nx \l_tmpa_tl {
1684     { \stex_term_math_assoc_arg:nnnn
1685       { \int_use:N \l_tmpa_int }
1686       { \l_tmpb_str }
1687       { \l_tmpa_cs {#####1} {#####2} }
1688       { ####\int_use:N \l_tmpa_int }
1689     }
1690   }
1691   \__stex_notation_arguments:
1692 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

1693 \cs_new_protected:Nn \__stex_notation_final: {
1694   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
1695   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
1696   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1697   \cs_generate_from_arg_count:cNnn {
1698     stex_notation_ \l_tmpa_str \c_hash_str
1699     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1700     _cs
1701   }
1702   \cs_set:Npx \l_tmpb_str {
1703     \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl
1704   }
1705
1706   \stex_debug:n{
1707     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1708     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
1709     Operator~precedence:~
1710     \prop_item:Nn \l_tmpb_prop { opprec }^^J
1711     Argument~precedences:~
1712     \seq_use:Nn \l_tmpa_seq {,~}^^J
1713     Notation: \cs_meaning:c {

```

```

1714     stex_notation_ \l_tmpa_str \c_hash_str
1715     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1716     _cs
1717 }
1718 }
1719
1720 \prop_gset_eq:cN {
1721     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1722     \c_hash_str \l__stex_notation_lang_str _prop
1723 } \l_tmpb_prop
1724
1725 \exp_args:Nx
1726 \stex_add_to_current_module:n {
1727     \prop_get:cnN {
1728         g_stex_symdecl_
1729         \prop_item:Nn \l_tmpb_prop { symbol }
1730         _prop
1731     } { notations } \exp_not:N \l_tmpa_seq
1732     \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
1733         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1734     }
1735     \prop_put:cno {
1736         g_stex_symdecl_
1737         \prop_item:Nn \l_tmpb_prop { symbol }
1738         _prop
1739     } { notations } \exp_not:N \l_tmpa_seq
1740 }
1741
1742 \stex_if_smsmode:TF {
1743     \stex_smsmode_set_codes:
1744     \exp_args:Nx \stex_addtosms:n {
1745         \prop_gset_from_keyval:cn {
1746             g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1747             \c_hash_str \l__stex_notation_lang_str _prop
1748         } {
1749             symbol      = \prop_item:Nn \l_tmpb_prop { symbol }
1750             language    = \prop_item:Nn \l_tmpb_prop { language }
1751             variant      = \prop_item:Nn \l_tmpb_prop { variant }
1752             opprec       = \prop_item:Nn \l_tmpb_prop { opprec }
1753             argprecs     = \prop_item:Nn \l_tmpb_prop { argprecs }
1754         }
1755     }
1756 }{
1757     \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
1758     \seq_put_right:Nx \l_tmpa_seq {
1759         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1760     }
1761     \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1762     \prop_set_eq:cN {
1763         g_stex_symdecl_ \l_tmpa_str _prop
1764     } \l_tmpa_prop
1765
1766     % HTML annotations
1767     \stex_annotate_invisible:nnn { notation }

```

```

1768 { \prop_item:Nn \l_tmpb_prop { symbol } } {
1769   \stex_annotate_invisible:nnn { notationfragment }
1770   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
1771   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1772   \stex_annotate_invisible:nnn { precedence }
1773   { \prop_item:Nn \l_tmpb_prop { opprec };
1774     \seq_use:Nn \l_tmpa_seq { x }
1775   }{}
1776
1777   \int_zero:N \l_tmpa_int
1778   \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1779   \tl_clear:N \l_tmpa_tl
1780   \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
1781     \int_incr:N \l_tmpa_int
1782     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1783     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1784     \str_if_eq:VnTF \l_tmpb_str a {
1785       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1786         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1787         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
1788       } }
1789     }{
1790       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1791         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
1792       } }
1793     }
1794   }
1795   \stex_annotate_invisible:nnn { notationcomp }{}{
1796     $ \exp_args:Nno \use:nn { \use:c {
1797       stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }
1798       \c_hash_str \l__stex_notation_variant_str
1799       \c_hash_str \l__stex_notation_lang_str _cs
1800     } } { \l_tmpa_tl } $
1801   }
1802 }
1803 }
1804 }

```

(End definition for _stex_notation_final:.)

\symdef

```

1805 \keys_define:nn { stex / symdef } {
1806   name .tl_set_x:N = \l_stex_symdecl_name_str ,
1807   local .bool_set:N = \l_stex_symdecl_local_bool ,
1808   args .tl_set_x:N = \l_stex_symdecl_args_str ,
1809   type .tl_set:N = \l_stex_symdecl_type_tl ,
1810   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1811   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1812   prec .tl_set_x:N = \l__stex_notation_prec_str ,
1813   unknown .code:n = \str_set:Nx
1814     \l__stex_notation_variant_str \l_keys_key_str
1815 }
1816
1817 \cs_new_protected:Nn \_stex_notation_symdef_args:n {

```



```

1818 \str_clear:N \l_stex_symdecl_name_str
1819 \str_clear:N \l_stex_symdecl_args_str
1820 \bool_set_false:N \l_stex_symdecl_local_bool
1821 \tl_clear:N \l_stex_symdecl_type_tl
1822 \str_clear:N \l__stex_notation_lang_str
1823 \str_clear:N \l__stex_notation_variant_str
1824 \str_clear:N \l__stex_notation_prec_str
1825
1826 \keys_set:nn { stex /symdef } { #1 }
1827
1828 \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
1829   \l_stex_symdecl_name_str
1830 \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
1831   \l_stex_symdecl_args_str
1832 \exp_args:NNo \str_set:Nn \l__stex_notation_lang_str
1833   \l__stex_notation_lang_str
1834 \exp_args:NNo \str_set:Nn \l__stex_notation_variant_str
1835   \l__stex_notation_variant_str
1836 \exp_args:NNo \str_set:Nn \l__stex_notation_prec_str
1837   \l__stex_notation_prec_str
1838 }
1839
1840 \NewDocumentCommand \symdef { O{} m } {
1841   \__stex_notation_symdef_args:n { #1 }
1842   \tl_clear:N \l_stex_symdecl_definiens_tl
1843   \bool_set_true:N \l_stex_symdecl_make_macro_bool
1844   \stex_symdecl_do:n { #2 }
1845   \exp_args:Nx \stex_notation_do:nn {
1846     \prop_item:Nn \l_tmpa_prop { module } ?
1847     \prop_item:Nn \l_tmpa_prop { name }
1848   }
1849 }

```

(End definition for `\symdef`. This function is documented on page 22.)

`\stex_invoke_symbol:n` Invokes a semantic macro

```

1850 \cs_new_protected:Nn \stex_invoke_symbol:n {
1851   \peek_charcode_remove:NTF ! {
1852     \stex_term_custom:nn { #1 } { }
1853   } {
1854     \if_mode_math:
1855       \exp_after:wN \__stex_notation_invoke_math:n
1856     \else:
1857       \exp_after:wN \__stex_notation_invoke_text:n
1858     \fi: { #1 }
1859   }
1860 }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 21.)

`__stex_notation_invoke_math:n`

```

1861 \cs_new_protected:Nn \__stex_notation_invoke_math:n {
1862   \peek_charcode_remove:NTF * {
1863     \__stex_notation_invoke_text:n { #1 }

```

```

1864 }{
1865   \peek_charcode:NTF [ {
1866     \__stex_notation_invoke_math:nw { #1 }
1867   }{
1868     \__stex_notation_invoke_math:nw { #1 } []
1869   }
1870 }
1871 }

```

(End definition for __stex_notation_invoke_math:n.)

_stex_notation_invoke_math:nw

```

1872 \cs_new_protected:Npn \__stex_notation_invoke_math:nw #1 [#2] {
1873   \__stex_notation_args:n { #2 }
1874   \prop_set_eq:Nc \l_tmpa_prop {
1875     g_stex_symdecl_ #1 _prop
1876   }
1877   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
1878   \seq_if_empty:NTF \l_tmpa_seq {
1879     \msg_set:nnn{stex}{error/nonotations}{
1880       Symbol~#1~used,~but~has~no~notations!
1881     }
1882     \msg_error:nn{stex}{error/nonotations}
1883   } {
1884     \seq_if_in:NxTF \l_tmpa_seq
1885       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
1886       \use:c{
1887         stex_notation_ #1 \c_hash_str
1888         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1889         _cs
1890       }
1891     }{
1892       \str_if_empty:NTF \l__stex_notation_variant_str {
1893         \str_if_empty:NTF \l__stex_notation_lang_str {
1894           \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
1895           \use:c{
1896             stex_notation_ #1 \c_hash_str \l_tmpa_str
1897             _cs
1898           }
1899         }{
1900           \msg_set:nnn{stex}{error/wrongnotation}{
1901             Symbol~#1~has~no~notation~
1902             \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1903           }
1904           \msg_error:nn{stex}{error/wrongnotation}
1905         }
1906       }{
1907         \msg_set:nnn{stex}{error/wrongnotation}{
1908           Symbol~#1~has~no~notation~
1909           \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1910         }
1911         \msg_error:nn{stex}{error/wrongnotation}
1912       }
1913     }

```

```

1914 }
1915 }

(End definition for \_stex_notation_invoke_math:nw.)

```

_stex_notation_invoke_text:n

```

1916 \cs_new_protected:Nn \_stex_notation_invoke_text:n {
1917   \prop_set_eq:Nc \l_tmpa_prop {
1918     g_stex_symdecl_ #1 _prop
1919   }
1920   \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1921   \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
1922 }

(End definition for \_stex_notation_invoke_text:n.)

```

4.8 Terms

```

1923 <@=stex_term>

```

Precedences:

```

\infprec
\neginfprec
\l__stex_term_downprec
1924 \int_const:Nn \infprec {\c_max_int}
1925 \int_const:Nn \neginfprec {-\c_max_int}
1926 \int_new:N \l__stex_term_downprec
1927 \int_set_eq:NN \l__stex_term_downprec \neginfprec

```

(End definition for \infprec, \neginfprec, and \l__stex_term_downprec. These variables are documented on page 23.)

Bracketing:

```

\l_stex_term_left_bracket_str
\l_stex_term_right_bracket_str
1928 \tl_set:Nn \l__stex_term_left_bracket_str (
1929 \tl_set:Nn \l__stex_term_right_bracket_str )
1930 \RequirePackage{scalereel}

```

(End definition for \l_stex_term_left_bracket_str and \l_stex_term_right_bracket_str.)

_stex_term_maybe_brackets:nn

Compares precedences and insert brackets accordingly

```

1931 \cs_new_protected:Nn \_stex_term_maybe_brackets:nn {
1932   \int_compare:nNnTF { #1 } < \l__stex_term_downprec {
1933     \STEXdobrackets { #2 }
1934   }{ #2 }
1935 }

```

(End definition for _stex_term_maybe_brackets:nn.)

\STEXdobrackets

```

1936 \cs_new_protected:Npn \STEXdobrackets #1 {
1937   \ThisStyle{if D\m@switch
1938     \exp_args:Nnx \use:nn
1939     { \left\l_stex_term_left_bracket_str #1 }
1940     { \right\l_stex_term_right_bracket_str }
1941   \else
1942     \exp_args:Nnx \use:nn

```

```

1943     { \l__stex_term_left_bracket_str #1 }
1944     { \l__stex_term_right_bracket_str }
1945   \fi}
1946 }

```

(End definition for `\STEXdobrackets`. This function is documented on page 23.)

`\STEXwithbrackets`

```

1947 \cs_new_protected:Npn \STEXwithbrackets #1 #2 #3 {
1948   \exp_args:Nnx \use:nn
1949   {
1950     \tl_set:Nx \l__stex_term_left_bracket_str { #1 }
1951     \tl_set:Nx \l__stex_term_right_bracket_str { #2 }
1952     #3
1953   }
1954   {
1955     \tl_set:Nn \exp_not:N \l__stex_term_left_bracket_str
1956     {\l__stex_term_left_bracket_str}
1957     \tl_set:Nn \exp_not:N \l__stex_term_right_bracket_str
1958     {\l__stex_term_right_bracket_str}
1959   }
1960 }

```

(End definition for `\STEXwithbrackets`. This function is documented on page 23.)

OMDOC terms:

`_stex_term_math_oms:nnnn`

```

1961 \cs_new_protected:Nn \_stex_term_oms:nnn {
1962   \stex_annotate:nnn{ OMID }{ #2 }{
1963     \stex_highlight_term:nn { #1 } { #3 }
1964   }
1965 }
1966
1967 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
1968   \_stex_term_maybe_brackets:nn { #3 }{
1969     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
1970   }
1971 }

```

(End definition for `_stex_term_math_oms:nnnn`. This function is documented on page 22.)

`_stex_term_math_oma:nnnn`

```

1972 \cs_new_protected:Nn \_stex_term_oma:nnn {
1973   \stex_annotate:nnn{ OMA }{ #2 }{
1974     \stex_highlight_term:nn { #1 } { #3 }
1975   }
1976 }
1977
1978 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
1979   \_stex_term_maybe_brackets:nn { #3 }{
1980     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
1981   }
1982 }

```

(End definition for `_stex_term_math_oma:nnnn`. This function is documented on page 22.)

`_stex_term_math_omb:nnnn`

```

1983 \cs_new_protected:Nn \_stex_term_ombind:nnn {
1984   \stex_annotate:nnn{ OMBIND }{ #2 }{
1985     \stex_highlight_term:nn { #1 } { #3 }
1986   }
1987 }
1988
1989 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
1990   \__stex_term_maybe_brackets:nn { #3 }{
1991     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
1992   }
1993 }

```

(End definition for `_stex_term_math_omb:nnnn`. This function is documented on page 22.)

`_stex_term_math_arg:nnn`

```

1994 \cs_new_protected:Nn \_stex_term_arg:nn {
1995   \stex_unhighlight_term:n {
1996     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
1997   }
1998 }
1999 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2000   \exp_args:Nnx \use:nn
2001   { \int_set:Nn \l__stex_term_downprec { #2 }
2002     \_stex_term_arg:nn { #1 } { #3 }
2003   }
2004   { \int_set:Nn \l__stex_term_downprec { \int_use:N \l__stex_term_downprec } }
2005 }

```

(End definition for `_stex_term_math_arg:nnn`. This function is documented on page 23.)

`_stex_term_math_assoc_arg:nnnn`

```

2006 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2007   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
2008   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
2009     \tl_set:Nn \l_tmpa_tl { #4 }
2010   }{
2011     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2012     \seq_reverse:N \l_tmpa_seq
2013     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
2014     \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
2015     \seq_map_inline:Nn \l_tmpa_seq {
2016       \tl_set:Nx \l_tmpa_tl {
2017         \exp_args:Nno
2018         \l_tmpa_cs { ##1 } { \l_tmpa_tl }
2019       }
2020     }
2021   }
2022   \exp_args:Nnno
2023   \_stex_term_math_arg:nnn{#1}{#2}{ \l_tmpa_tl }
2024 }

```

(End definition for `_stex_term_math_assoc_arg:nnnn`. This function is documented on page 23.)

`\stex_term_custom:nn`

```

2025 \cs_new_protected:Nn \stex_term_custom:nn {
2026   \str_set:Nn \l__stex_term_custom_uri { #1 }
2027   \str_set:Nn \l_tmpa_str { #2 }
2028   \tl_clear:N \l_tmpa_tl
2029   \int_zero:N \l_tmpa_int
2030   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2031   \__stex_term_custom_loop:
2032 }

```

(End definition for `\stex_term_custom:nn`. This function is documented on page 24.)

`__stex_term_custom_loop:`

```

2033 \cs_new_protected:Nn \__stex_term_custom_loop: {
2034   \bool_set_false:N \l_tmpa_bool
2035   \bool_while_do:nn {
2036     \str_if_eq_p:ee X {
2037       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2038     }
2039   }{
2040     \int_incr:N \l_tmpa_int
2041   }
2042
2043   \peek_charcode:NTF [ {
2044     % notation/text component
2045     \__stex_term_custom_component:w
2046   } {
2047     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2048       % all arguments read => finish
2049       \__stex_term_custom_final:
2050     } {
2051       % arguments missing
2052       \peek_charcode_remove:NTF * {
2053         % invisible, specific argument position or both
2054         \peek_charcode:NTF [ {
2055           % visible specific argument position
2056           \__stex_term_custom_arg:wn
2057         } {
2058           % invisible
2059           \peek_charcode_remove:NTF * {
2060             % invisible specific argument position
2061             \__stex_term_custom_arg_inv:wn
2062           } {
2063             % invisible next argument
2064             \__stex_term_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2065           }
2066         }
2067       } {
2068         % next normal argument
2069         \__stex_term_custom_arg:wn [ \l_tmpa_int + 1 ]
2070       }
2071     }
2072   }
2073 }

```

(End definition for _stex_term_custom_loop:.)

_stex_term_custom_arg_inv:wn

```

2074 \cs_new_protected:Npn \_stex_term_custom_arg_inv:wn [ #1 ] #2 {
2075   \bool_set_true:N \l_tmpa_bool
2076   \_stex_term_custom_arg:wn [ #1 ] { #2 }
2077 }

```

(End definition for _stex_term_custom_arg_inv:wn.)

_stex_term_custom_arg:wn

```

2078 \cs_new_protected:Npn \_stex_term_custom_arg:wn [ #1 ] #2 {
2079   \str_set:Nx \l_tmpb_str {
2080     \str_item:Nn \l_tmpa_str { #1 }
2081   }
2082   \str_case:VnTF \l_tmpb_str {
2083     { X } { } % TODO throw error
2084     { i } { \_stex_term_custom_set_X:n { #1 } }
2085     { b } { \_stex_term_custom_set_X:n { #1 } }
2086     { a } { } % TODO ?
2087   }{}{
2088     % TODO throw error
2089   }
2090
2091   \bool_if:nTF \l_tmpa_bool {
2092     \tl_put_right:Nx \l_tmpa_tl {
2093       \stex_annotate_invisible:n {
2094         \_stex_term_arg:nn { \int_eval:n { #1 } }
2095         \exp_not:n { { #2 } }
2096       }
2097     }
2098   } {
2099     \tl_put_right:Nx \l_tmpa_tl {
2100       \_stex_term_arg:nn { \int_eval:n { #1 } }
2101       \exp_not:n { { #2 } }
2102     }
2103   }
2104
2105   \_stex_term_custom_loop:
2106 }

```

(End definition for _stex_term_custom_arg:wn.)

_stex_term_custom_set_X:n

```

2107 \cs_new_protected:Nn \_stex_term_custom_set_X:n {
2108   \str_set:Nx \l_tmpa_str {
2109     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2110     X
2111     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2112   }
2113 }

```

(End definition for _stex_term_custom_set_X:n.)

_stex_term_custom_component:

```

2114 \cs_new_protected:Npn \_stex_term_custom_component:w [ #1 ] {
2115   \tl_put_right:Nn \l_tmpa_tl { #1 }
2116   \_stex_term_custom_loop:
2117 }

```

(End definition for _stex_term_custom_component:.)

_stex_term_custom_final:

```

2118 \cs_new_protected:Nn \_stex_term_custom_final: {
2119   \int_compare:nNnTF \l_tmpb_int = 0 {
2120     \exp_args:Nnno \_stex_term_oms:nnn
2121   }{
2122     \str_if_in:NnTF \l_tmpa_str {b} {
2123       \exp_args:Nnno \_stex_term_ombind:nnn
2124     } {
2125       \exp_args:Nnno \_stex_term_oma:nnn
2126     }
2127   }
2128   { \l__stex_term_custom_uri } { \l__stex_term_custom_uri } { \l_tmpa_tl }
2129 }

```

(End definition for _stex_term_custom_final:.)

\stex_highlight_term:nn

```

2130 \latexml_if:F {
2131   \scalatex_if:F{
2132     \RequirePackage{pdfcomment}
2133   }
2134 }
2135
2136 \str_new:N \l__stex_term_highlight_uri_str
2137 \cs_new_protected:Nn \stex_highlight_term:nn {
2138   \latexml_if:TF {
2139     #2
2140   } {
2141     \scalatex_if:TF {
2142       #2
2143     } {
2144       \exp_args:Nnx
2145       \use:nn {
2146         \str_set:Nx \l__stex_term_highlight_uri_str { #1 }
2147         #2
2148       } {
2149         \str_set:Nx \exp_not:N \l__stex_term_highlight_uri_str
2150         { \l__stex_term_highlight_uri_str }
2151       }
2152     }
2153   }
2154 }
2155
2156 \cs_new_protected:Nn \stex_unhighlight_term:n {
2157   % \latexml_if:TF {
2158   %   #1

```



```

2159 % } {
2160 % \scalatex_if:TF {
2161 % #1
2162 % } {
2163 % #1 %\iffalse{\fi}} #1 {\iffalse}}\fi
2164 % }
2165 % }
2166 }

```

(End definition for `\stex_highlight_term:nn`. This function is documented on page 24.)

```

\comp
\@comp
2167 \cs_new_protected:Npn \comp #1 {
2168   \str_if_empty:NF \l__stex_term_highlight_uri_str {
2169     \exp_args:Nnx \@comp { #1 } { \l__stex_term_highlight_uri_str }
2170   }
2171 }
2172
2173 \cs_new_protected:Npn \@comp #1 #2 {
2174   \pdfxtooltip {
2175     \textcolor{blue}{#1}
2176   } { #2 }
2177 }

```

(End definition for `\comp` and `\@comp`. These functions are documented on page 24.)

```

2178 \@ifpackageloaded{tikzinput}{
2179   \RequirePackage{stex-tikzinput}
2180 }{}
2181 \</package>

```

4.9 Auxiliary Packages

4.9.1 tikzinput

```

2182 \< *tikzinput>
2183 \< @@=tikzinput>
2184 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
2185 \RequirePackage{13keys2e}
2186
2187 \keys_define:nn { tikzinput } {
2188   image .bool_set:N = \c_tikzinput_image_bool
2189 }
2190
2191 \ProcessKeysOptions { tikzinput }
2192
2193 \bool_if:NTF \c_tikzinput_image_bool {
2194   \RequirePackage{graphicx}
2195
2196   \providecommand\usetikzlibrary[]{}
2197   \newcommand\tikzinput[2][] {\includegraphics[#1]{#2}}
2198 }{
2199   \RequirePackage{tikz}
2200   \RequirePackage{standalone}
2201 }

```

```

2202 \newcommand \tikzinput [2] [] {
2203   \setkeys{Gin}{#1}
2204   \ifx \Gin@width \Gin@exclamation
2205     \ifx \Gin@height \Gin@exclamation
2206       \input { #2 }
2207     \else
2208       \resizebox{!}{ \Gin@height }{
2209         \input { #2 }
2210       }
2211     \fi
2212   \else
2213     \ifx \Gin@height \Gin@exclamation
2214       \resizebox{ \Gin@width }{!}{
2215         \input { #2 }
2216       }
2217     \else
2218       \resizebox{ \Gin@width }{ \Gin@height }{
2219         \input { #2 }
2220       }
2221     \fi
2222   \fi
2223 }
2224 }
2225
2226 \newcommand \ctikzinput [2] [] {
2227   \begin{center}
2228     \tikzinput [#1] {#2}
2229   \end{center}
2230 }
2231
2232 \@ifpackageloaded{stex}{
2233   \RequirePackage{stex-tikzinput}
2234 }{}
2235 </tikzinput>
2236 < *stex-tikzinput>
2237 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
2238 \RequirePackage{stex}
2239 \RequirePackage{tikzinput}
2240
2241 % TODO
2242
2243 </stex-tikzinput>

```

4.9.2 sTeX1 Compatibility

```

2244 < *smglom>
2245 \RequirePackage{expl3,l3keys2e}
2246 \ProvidesExplClass{smglom}{2021/08/01}{1.9}{sTeX1 compatibility}
2247 \LoadClass[border=1px,varwidth]{standalone}
2248 \setlength\textwidth{15cm}
2249 %\g@addto@macro{\@parboxrestore}{\setlength\parskip{\baselineskip}}
2250 \DeclareOption{mh}{}
2251 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
2252 \ProcessOptions
2253

```

```

2254 \RequirePackage{stex-compatibility}
2255 \</smglom>
2256
2257 <*compat>
2258 \ProvidesExplPackage{stex-compatibility}{2021/08/01}{1.9}{bla}
2259 \RequirePackage[debug,lang={de,en}]{stex}
2260
2261 \NewDocumentEnvironment { mhmodnl } { 0{} m m } {
2262   \msg_set:nnn{stex}{warning/deprecated}{
2263     \\\
2264     Environment~mhmodnl~is~deprected! \\\
2265     Please~update~module~#2~in~file~
2266     \stex_path_to_string:N \g_stex_currentfile_seq!
2267     \\\ \\\
2268   }
2269   \msg_warning:nn{stex}{warning/deprecated}
2270
2271   \begin{module}[#1,lang=#3]{#2}
2272     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2273     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2274     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
2275     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
2276     \input { \stex_path_to_string:N \l_tmpa_seq / \l_tmpa_str.tex }
2277   } {
2278     \end{module}
2279   }
2280
2281 \NewDocumentEnvironment { modsig } { 0{} m } {
2282   \stex_if_in_module:TF {
2283     \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str
2284     \str_set:Nn \l_tmpb_str { #2 }
2285     \str_if_eq:NNTF \l_tmpa_str \l_tmpb_str {
2286       \prop_set_eq:NN \l_modsig_old_module_prop \l_stex_current_module_prop
2287       \begin{@module}{modsig-#2}
2288         % \prop_set_eq:NN \l_stex_current_module_prop \l_modsig_old_module_prop
2289       } {
2290         \begin{@module}{#2}
2291       }
2292     } {
2293       \begin{@module}{#2}
2294     }
2295   }{
2296     \end{@module}
2297     \AddToHookNext { env / modsig / after }{
2298       \stex_if_in_module:T {
2299         \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str
2300         \str_set:Nn \l_tmpb_str { #2 }
2301         \str_if_eq:NNT \l_tmpa_str \l_tmpb_str {
2302           % \xdef \g_stex_module_after_group_tl {
2303             \stex_if_smsmode:TF {
2304               \exp_args:Nx
2305               \stex_add_to_current_module:n {
2306                 \stex_debug:n{Activating~signature~of~#2}
2307                 \exp_not:N \prop_item:cn { c_stex_module_

```

```

2308         \prop_item:Nn \l_stex_current_module_prop {ns} ?
2309         \prop_item:Nn \l_stex_current_module_prop {name}
2310         / modsig-#2_prop } { content }
2311     }
2312 }
2313 {
2314     \gdef \g_stex_module_after_group_tl {
2315         \stex_debug:n{Activating~signature~of~#2}
2316
2317         \seq_put_right:Nx \l_stex_all_modules_seq {
2318             \prop_item:Nn \l_stex_current_module_prop {ns} ?
2319             \prop_item:Nn \l_stex_current_module_prop {name}
2320             / modsig-#2_prop
2321         }
2322         \prop_item:cn { c_stex_module_
2323         \prop_item:Nn \l_stex_current_module_prop {ns} ?
2324         \prop_item:Nn \l_stex_current_module_prop {name}
2325         / modsig-#2_prop } { content }
2326         \exp_args:Nx
2327         \stex_add_to_current_module:n {
2328             \stex_debug:n{Activating~signature~of~#2}
2329             \exp_not:N \prop_item:cn { c_stex_module_
2330             \prop_item:Nn \l_stex_current_module_prop {ns} ?
2331             \prop_item:Nn \l_stex_current_module_prop {name}
2332             / modsig-#2_prop } { content }
2333         }
2334     }
2335     \aftergroup \g_stex_module_after_group_tl
2336 }
2337 % }
2338 % \aftergroup \g_stex_module_after_group_tl
2339 }
2340 }
2341 }
2342 }
2343
2344 \NewDocumentCommand \gimport { 0{} m } {
2345     \msg_set:nnn{stex}{warning/deprecated}{
2346         \\\
2347         \c_backslash_str gimport~is~deprecated! \\\
2348         Please~use~\c_backslash_str importmodule[#1]{#2}~instead!~(in~file~
2349         \stex_path_to_string:N \g_stex_currentfile_seq)
2350         \\\ \\\
2351     }
2352     \msg_warning:nn{stex}{warning/deprecated}
2353     \importmodule[#1]{#2}
2354 }
2355
2356 \cs_new_protected:Npn \symi {
2357     \peek_charcode_remove:NTF * {
2358         \symi_do:
2359     } {
2360         \symi_do:
2361     }

```

```

2362 }
2363
2364 \NewDocumentCommand \symi_do: { 0{ } m } {
2365   \msg_set:nnn{stex}{warning/deprecated}{
2366     \\\
2367     \c_backslash_str symi~is~deprecated! \\\
2368     Please~use~\c_backslash_str symdecl{#1}{#2}~instead!~(in~file~
2369     \stex_path_to_string:N \g_stex_currentfile_seq)
2370     \\\ \\\
2371   }
2372   \msg_warning:nn{stex}{warning/deprecated}
2373   \symdecl*{#1}{#2}
2374 }
2375
2376 \cs_new_protected:Npn \symii {
2377   \peek_charcode_remove:NTF * {
2378     \symii_do:
2379   } {
2380     \symii_do:
2381   }
2382 }
2383
2384 \NewDocumentCommand \symii_do: { 0{ } m m } {
2385   \msg_set:nnn{stex}{warning/deprecated}{
2386     \\\
2387     \c_backslash_str symii~is~deprecated! \\\
2388     Please~use~\c_backslash_str symdecl{#1}{#2-#3}~instead!~(in~file~
2389     \stex_path_to_string:N \g_stex_currentfile_seq)
2390     \\\ \\\
2391   }
2392   \msg_warning:nn{stex}{warning/deprecated}
2393   \symdecl*{#1}{#2-#3}
2394 }
2395
2396 \cs_new_protected:Npn \defi {
2397   \peek_charcode_remove:NTF * {
2398     \defi_do:
2399   } {
2400     \defi_do:
2401   }
2402 }
2403
2404 \NewDocumentCommand \defi_do: { 0{ } m } {
2405   \str_set:Nn \l_tmpa_str { #1 }
2406   \str_if_empty:NTF \l_tmpa_str {
2407     \msg_set:nnn{stex}{warning/deprecated}{
2408       \\\
2409       \c_backslash_str defi~is~deprecated! \\\
2410       Please~use~\c_backslash_str STEXsymbol{#2}!{#2}~instead!~(in~file~
2411       \stex_path_to_string:N \g_stex_currentfile_seq)
2412       \\\ \\\
2413     }
2414     \msg_warning:nn{stex}{warning/deprecated}
2415     \STEXsymbol { #2 }![ \comp{#2} ]

```

```

2416 } {
2417   \msg_set:nnn{stex}{warning/deprecated}{
2418     \\\
2419     \c_backslash_str defii-is-deprecated! \\\
2420     Please~use~\c_backslash_str STExsymbol { #1 ? #2 }[ #2 ]~instead!~(in~file~
2421     \stex_path_to_string:N \g_stex_currentfile_seq)
2422     \\\
2423   }
2424   \msg_warning:nn{stex}{warning/deprecated}
2425   \STExsymbole { #1 ? #2 }[ \comp{#2} ]
2426 }
2427 }
2428
2429 \cs_new_protected:Npn \defii {
2430   \peek_charcode_remove:NTF * {
2431     \defii_do:
2432   } {
2433     \defii_do:
2434   }
2435 }
2436
2437 \NewDocumentCommand \defii_do: { 0{} m m } {
2438   \str_set:Nn \l_tmpa_str { #1 }
2439   \str_if_empty:NTF \l_tmpa_str {
2440     \msg_set:nnn{stex}{warning/deprecated}{
2441       \\\
2442       \c_backslash_str defii-is-deprecated! \\\
2443       Please~use~\c_backslash_str STExsymbol{#2-#3}![#2~#3]~instead!~(in~file~
2444       \stex_path_to_string:N \g_stex_currentfile_seq)
2445       \\\
2446     }
2447     \msg_warning:nn{stex}{warning/deprecated}
2448     \STExsymbol { #2-#3 }![ \comp{#2~#3} ]
2449   } {
2450     \msg_set:nnn{stex}{warning/deprecated}{
2451       \\\
2452       \c_backslash_str defii-is-deprecated! \\\
2453       Please~use~\c_backslash_str STExsymbol { #1 ? #2-#3 }[ #2~#3 ]~instead!~(in~file~
2454       \stex_path_to_string:N \g_stex_currentfile_seq)
2455       \\\
2456     }
2457     \msg_warning:nn{stex}{warning/deprecated}
2458     \STExsymbol { #1 ? #2-#3 }[ \comp{#2~#3} ]
2459   }
2460 }
2461
2462 %\RequirePackage[hyperref]{ntheorem}
2463 %\theoremstyle{plain}
2464 %\RequirePackage{amsthm}
2465
2466 \NewDocumentEnvironment {definition} { 0{} } {
2467   \stex_smsmode_set_codes:
2468   \msg_set:nnn{stex}{warning/deprecated}{
2469     \\\

```

```

2470     definition~environment~is~deprecated!~(in~file~
2471     \stex_path_to_string:N \g_stex_currentfile_seq)
2472     \\ \\
2473   }
2474   \msg_warning:nn{stex}{warning/deprecated}
2475 }{}
2476
2477 \NewDocumentCommand \trefi { 0{} m } {
2478   \str_set:Nn \l_tmpa_str { #1 }
2479   \str_if_empty:NTF \l_tmpa_str {
2480     \msg_set:nnn{stex}{warning/deprecated}{
2481       \\
2482       \c_backslash_str trefi~is~deprecated! \\
2483       Please~use~\c_backslash_str STEXsymbol{#2}! [#2]~instead!~(in~file~
2484       \stex_path_to_string:N \g_stex_currentfile_seq)
2485       \\ \\
2486     }
2487     \msg_warning:nn{stex}{warning/deprecated}
2488     \STEXsymbol { #2 }! [ \comp{#2} ]
2489   } {
2490     \msg_set:nnn{stex}{warning/deprecated}{
2491       \\
2492       \c_backslash_str trefi~is~deprecated! \\
2493       Please~use~\c_backslash_str STEXsymbol { #1 ? #2 } [ #2 ]~instead!~(in~file~
2494       \stex_path_to_string:N \g_stex_currentfile_seq)
2495       \\ \\
2496     }
2497     \msg_warning:nn{stex}{warning/deprecated}
2498     \STEXsymbol { #1 ? #2 } [ \comp{#2} ]
2499   }
2500 }
2501
2502
2503 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { mhmodnl } }
2504 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { modsig } }
2505 \tl_gput_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\gimport\syml\symii\symiii\symiv}
2506
2507 </compat>

```