

stex.sty: \TeX 2.0*

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

2021-09-04

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	68
4.9	Auxiliary Packages	75

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}]*{$\mult{a}{b}$}[\comp{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}
$\numseq{a,b,c}{\mathbb R}$
```

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

Finally, B-type arguments combine the functionalities of a and b, i.e. they represent flexary binding operator arguments.

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 $\texttt{\textbackslash foo}$ 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 $\texttt{\textbackslash foo}$), its operator precedence (e.g. 200) is compared to the current downwards precedence (initially $\texttt{\textbackslash neginfpred}$). 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 $\texttt{\textbackslash infpred}$, 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. $\int \int \int f \, dx \, dy \, dz$?

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

2.3 Archives and Imports

2.3.1 Namespaces

Ideally, \S T E X 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, \T E X 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 T E X 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 $\langle top-directory \rangle / some/path / Foo[.⟨lang⟩].tex$, or in $\langle top-directory \rangle / some/path[.⟨lang⟩].tex$ (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 / .. / ..} & \\
\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`
`\STEXexport`

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 <code>module-environment</code> 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`

`\stex_notation_do:nn`

`\stex_notation_do:nn{<URI>}{<notations+>}`

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

Ultimately stores the notation in the property list `\g_stex_notation_<URI>#<variant>#<lang>_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}_ {\com
\end{module}
```

Module 3.13[NotationTest]

`\symdef`

`\symdef[<args>]{<symbol>}{<notations+>}`

Combines `\symdecl` and `\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]
 $a+b+c$

`_stex_term_math_oms:nnnn`
`_stex_term_math_oma:nnnn`
`_stex_term_math_omb:nnnn`

`<URI><fragment><precedence><body>`

Annotates `<body>` as an OMDOC-term (OMID, OMA or OMBIND, respectively) with head symbol `<URI>`, generated by the specific notation `<fragment>` with (upwards) operator precedence `<precedence>`. Inserts parentheses according to the current downwards precedence and operator precedence.

<div> <div>Module 3.16[MathTest2]</div> <div> $\langle a _{[b,c,d,e,f]}^g \rangle$ and $\langle a _{[b,c]}^g \rangle$ and $\langle a _{[b]}^c \rangle$ $a+b \cdot c$ and $a \cdot (\frac{a}{b} + \frac{a}{c})$ $a+b \cdot c$ and $a \cdot \left(\frac{a}{b} + \frac{a}{c} \right)$ $a+b \cdot c$ and $a \cdot \left[\frac{a}{b} + \frac{a}{c} \right]$ </div> </div>
--

<u><u><code>\stex_term_custom:nn</code></u></u>	<code>\stex_term_custom:nn{<URI>}{<args>}</code>	Implements custom one-time notation. Invoked by <code>\stex:invoke_symbol:n</code> in text mode, or if followed by <code>*</code> in math mode, or whenever followed by <code>!</code> .
---	--	--

<div> <div>Test 16</div> <div> <pre> \begin{module}{TextTest} \importmodule{Foo} \bar[some]a[and some]b[and also some]c[here]. \$\bar*[\text{some }]a[\text{ and some }]b[\text{ and also some }]c[\text{ here}]\$. \$\bar![\mathtt{bar}]\$ \bar*{a}*{b}[or just some]c \bar![bar] \bar[or first]*[2]{b}[, then]*[3]{c}[, and finally]a \end{module} </pre> </div> </div>
<div> <div>Module 3.17[TextTest]</div> <div> some a and some b and also some c here. some a and some b and also some c here. bar or just some c bar or first b, then c, and finally a </div> </div>

<u><u><code>\stex_highlight_term:nn</code></u></u>	<code>\stex_highlight_term:nn{<URI>}{<args>}</code>	Establishes a context for <code>\comp</code> . Stores the URI in a variable so that <code>\comp</code> knows which symbol governs the current notation.
--	---	---

<u><u><code>\comp</code></u></u>	<code>\comp{<args>}</code>	
<u><u><code>\@comp</code></u></u>		Marks <code><args></code> as a notation component of the current symbol for highlighting, linking, etc.

The precise behavior is governed by `\@comp`, which takes as additional argument the URI of the current symbol. By default, `\@comp` adds the URI as a PDF tooltip and colors the highlighted part in blue.

<u><u><code>\STEXinvisible</code></u></u>	Exports its argument as OMDOC (invisible), but does not produce PDF output. Useful e.g. for semantic macros that take arguments that are not part of the symbolic notation.
---	---

`\ellipses` `TODO`

4 Implementation

4.1 The `sTeX` 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 `sTeX` logo:

```
25 \protected\def\sTeX{%
26   \@ifundefined{texorpdfstring}%
27   {\let\texorpdfstring\@firstoftwo}%
28   }%
29   \texorpdfstring{\raisebox{-.5ex}{S\kern-.5exTeX}{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}{}
38
```

\stex_debug:n Debug mode

```

38 \cs_new_protected:Nn \stex_debug:n {
39   \bool_if:nT{\c_stex_debug_bool}{
40     \exp_args:Nnnx\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 <@=stex_annotate>

```

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

```

\c__stex_annotate_emptyarg_tl
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 `\l__stex_annotate_arg_tl` and `\c__stex_annotate_emptyarg_tl`.)

```

\__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 `__stex_annotate_checkempty:n`.)

```

\stex_annotate:nnx
\stex_annotate_invisible:n
\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_LA_TE_X, p_DF_LA_TE_X).

The p_DF_LA_TE_X-macros largely do nothing; the S_CA_LA_TE_X-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
\c__stex_path_up_str

```

```

. and .., respectively.
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:NNTF \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:N \underline{TF}

```

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:N\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:N\g__stex_files_stack{
325     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
326   }
327   \seq_if_empty:N\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:N\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:N\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:N\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`
`\STEXexport`

```

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 }
538 \NewDocumentCommand \STEXexport { m }{
539   \stex_smsmode_set_codes:
540   \stex_add_to_current_module:n { #1 }
541   #1
542 }

```

(End definition for `\stex_add_to_current_module:n` and `\STEXexport`. These functions are documented on page 12.)

`\stex_add_constant_to_current_module:n`

```

543 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
544   \str_set:Nx \l_tmpa_str { #1 }
545   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
546   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
547   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
548 }

```

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

`\stex_add_import_to_current_module:n`

```

549 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
550   \str_set:Nx \l_tmpa_str { #1 }
551   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
552   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
553   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
554 }

```

(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 555 \str_new:N \l_stex_modules_ns_str

```

```

556 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
557   \str_set:Nx \l_tmpa_str { #1 }
558   \seq_set_eq:NN \l_tmpa_seq #2
559   % split off file extension
560   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
561   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
562   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
563   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
564
565   \bool_set_true:N \l_tmpa_bool
566   \bool_while_do:Nn \l_tmpa_bool {
567     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
568     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
569       {source} { \bool_set_false:N \l_tmpa_bool }
570     }{}{
571       \seq_if_empty:NT \l_tmpa_seq {
572         \bool_set_false:N \l_tmpa_bool
573       }
574     }
575   }
576
577   \seq_if_empty:NTF \l_tmpa_seq {
578     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
579   }{
580     \str_set:Nx \l_stex_modules_ns_str {
581       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
582     }
583   }
584 }

```

(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:`

```

585 \cs_new_protected:Nn \stex_modules_current_namespace: {
586   \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
587     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
588   }{
589     % split off file extension
590     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
591     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
592     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
593     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
594     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
595     \str_set:Nx \l_stex_modules_ns_str {
596       file:/\stex_path_to_string:N \l_tmpa_seq
597     }
598   }
599 }

```

(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

```
600 \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
```

```
601 \NewDocumentCommand \STEXModule { m } {
602   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
603   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
604   \tl_set:Nn \l_tmpa_tl {
605     \msg_set:nnn{stex}{error/unknownmodule}{
606       No~module~#1~found!
607     }
608     \msg_error:nn{stex}{error/unknownmodule}
609   }
610   \seq_map_inline:Nn \l_stex_all_modules_seq {
611     \str_set:Nn \l_tmpb_str { ##1 }
612     \str_if_eq:eeT { \l_tmpa_str } {
613       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
614     } {
615       \seq_map_break:n {
616         \tl_set:Nn \l_tmpa_tl {
617           \stex_invoke_module:n { ##1 }
618         }
619       }
620     }
621   }
622   \l_tmpa_tl
623 }
624
625 \cs_new_protected:Nn \stex_invoke_module:n {
626   \stex_debug:n{Invoking~module~#1}
627   \peek_charcode_remove:NTF ! {
628     \__stex_module_invoke_uri:nN { #1 }
629   } {
630     \peek_charcode_remove:NTF ? {
631       \__stex_module_invoke_symbol:nn { #1 }
632     } {
633       \msg_set:nnn{stex}{error/syntax}{
634         Syntax~error:~?~or~!~expected~after~
635         \c_backslash_str STEXModule{#1}
636       }
637       \msg_error:nn{stex}{error/syntax}
638     }
639   }
640 }
641
642 \cs_new_protected:Nn \__stex_module_invoke_uri:nN {
643   \str_set:Nn #2 { #1 }
644 }
645
646 \cs_new_protected:Nn \__stex_module_invoke_symbol:nn {
647   \stex_invoke_symbol:n{#1?#2}
648 }
```


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

module module arguments:

```

649 \keys_define:nn { stex / module } {
650   title      .tl_set_x:N = \l_stex_module_title_str ,
651   ns         .tl_set_x:N = \l_stex_module_ns_str ,
652   lang       .tl_set_x:N = \l_stex_module_lang_str ,
653   sig        .tl_set_x:N = \l_stex_module_sig_str ,
654   creators   .tl_set_x:N = \l_stex_module_creators_str ,
655   contributors .tl_set_x:N = \l_stex_module_contributors_str ,
656   meta       .tl_set_x:N = \l_stex_module_meta_str
657 }
658
659 % module parameters here? In the body?
660
661 \cs_new_protected:Nn \__stex_module_args:n {
662   \str_clear:N \l_stex_module_title_str
663   \str_clear:N \l_stex_module_ns_str
664   \str_clear:N \l_stex_module_lang_str
665   \str_clear:N \l_stex_module_sig_str
666   \str_clear:N \l_stex_module_creators_str
667   \str_clear:N \l_stex_module_contributors_str
668   \str_clear:N \l_stex_module_meta_str
669   \keys_set:nn { stex / module } { #1 }
670   \exp_args:NNo \str_set:Nn \l_stex_module_title_str
671     \l_stex_module_title_str
672   \exp_args:NNo \str_set:Nn \l_stex_module_ns_str
673     \l_stex_module_ns_str
674   \exp_args:NNo \str_set:Nn \l_stex_module_lang_str
675     \l_stex_module_lang_str
676   \exp_args:NNo \str_set:Nn \l_stex_module_sig_str
677     \l_stex_module_sig_str
678   \exp_args:NNo \str_set:Nn \l_stex_module_meta_str
679     \l_stex_module_meta_str
680   \exp_args:NNo \str_set:Nn \l_stex_module_creators_str
681     \l_stex_module_creators_str
682   \exp_args:NNo \str_set:Nn \l_stex_module_contributors_str
683     \l_stex_module_contributors_str
684 }

```

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

```

685 \cs_new_protected:Nn \__stex_module_begin_module: {
686   % Nested module?
687   \stex_if_in_module:TF {
688     % Nested module
689     \prop_get:NnN \l_stex_current_module_prop
690       { ns } \l_stex_module_ns_str
691     \str_set:Nx \l_stex_module_name_str {
692       \prop_item:Nn \l_stex_current_module_prop
693         { name } / \l_stex_module_name_str
694     }
695   }{
696     % not nested:

```

```

697 \str_if_empty:NT \l_stex_module_ns_str {
698   \stex_modules_current_namespace:
699   \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
700   \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
701     / {\l_stex_module_ns_str}
702   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
703   \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
704     \str_set:Nx \l_stex_module_ns_str {
705       \stex_path_to_string:N \l_tmpa_seq
706     }
707   }
708 }
709 }
710
711 % language
712 \str_if_empty:NF \l_stex_module_lang_str {
713   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
714   \l_tmpa_str {
715     \exp_args:Nx \selectlanguage { \l_tmpa_str }
716   } {
717     \msg_set:nnn{stex}{error/unknownlanguage}{
718       Unknown~language~\l_tmpa_str
719     }
720     \msg_error:nn{stex}{error/unknownlanguage}
721   }
722 }
723
724 % signature
725 \str_if_empty:NF \l_stex_module_sig_str {
726   \str_if_empty:NT \l_stex_module_lang_str {
727     \msg_set:nnn{stex}{error/siglanguage}{
728       Module~\l_stex_module_ns_str?\l_stex_module_name_str~
729       declares~signature~\l_stex_module_sig_str,~but~does~not~
730       declare~its~language
731     }
732     \msg_error:nn{stex}{error/siglanguage}
733   }
734 }
735
736 % metatheory
737 % \str_if_empty:NTF \l_stex_module_meta_str {
738 %
739 % } {
740 %
741 % }
742
743 \str_clear:N \l_tmpa_str
744 \seq_clear:N \l_tmpa_seq
745 \tl_clear:N \l_tmpa_tl
746 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
747   name      = \l_stex_module_name_str ,
748   ns        = \l_stex_module_ns_str ,
749   imports    = \exp_not:o { \l_tmpa_seq } ,
750   constants = \exp_not:o { \l_tmpa_seq } ,

```

```

751     content    = \exp_not:o { \l_tmpa_tl } ,
752     file       = \exp_not:o { \g_stex_currentfile_seq } ,
753     lang       = \l_stex_module_lang_str ,
754     sig        = \l_stex_module_sig_str ,
755     meta       = \l_stex_module_meta_str
756 }
757
758 \stex_debug:n{
759   New~module:\\
760   Namespace:~\l_stex_module_ns_str\\
761   Name:~\l_stex_module_name_str\\
762   Language:~\l_stex_module_lang_str\\
763   Signature:~\l_stex_module_sig_str\\
764   Metatheory:~\l_stex_module_meta_str\\
765   File:~\stex_path_to_string:N \g_stex_currentfile_seq
766 }
767
768 \seq_put_right:Nx \l_stex_all_modules_seq {
769   \l_stex_module_ns_str ? \l_stex_module_name_str
770 }
771
772 \seq_gput_right:Nx \g_stex_modules_in_file_seq
773   { \l_stex_module_ns_str ? \l_stex_module_name_str }
774
775 \stex_if_smsmode:TF {
776   \stex_smsmode_set_codes:
777 } {
778   \begin{stex_annotate_env} {theory} {
779     \l_stex_module_ns_str ? \l_stex_module_name_str
780   }
781
782   \stex_annotate_invisible:nnn{header}{} {
783     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
784     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
785     \str_if_empty:NT \l_stex_module_meta_str {
786       % TODO metatheory
787     }
788   }
789 }
790 }
791 \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}

```

792 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
793 \cs_new_protected:Nn \_stex_module_end_module: {
794   \str_set:Nx \l_tmpa_str {
795     c_stex_module_
796     \prop_item:Nn \l_stex_current_module_prop { ns } ?
797     \prop_item:Nn \l_stex_current_module_prop { name }
798     _prop
799   }
800   %^^A \prop_new:c { \l_tmpa_str }

```

```

801 \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
802 \stex_debug:n{Closing~module~\prop_item:Nn \l_stex_current_module_prop { name }}
803 \stex_if_smsmode:TF {
804   \exp_args:Nx \stex_addtosms:n {
805     \prop_gset_from_keyval:cn {
806       c_stex_module_
807       \prop_item:Nn \l_stex_current_module_prop { ns } ?
808       \prop_item:Nn \l_stex_current_module_prop { name }
809       _prop
810     } {
811       name      = \prop_item:cn { \l_tmpa_str } { name } ,
812       ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
813       imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
814       constants = \prop_item:cn { \l_tmpa_str } { constants } ,
815       content   = \prop_item:cn { \l_tmpa_str } { content } ,
816       file      = \prop_item:cn { \l_tmpa_str } { file } ,
817       lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
818       sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
819       meta      = \prop_item:cn { \l_tmpa_str } { meta }
820     }
821   }
822 }{
823   \end{stex_annotate_env}
824 }
825 }

```

(End definition for `_stex_module_end_module:.`)

@module The core environment, with no header

```

826 \NewDocumentEnvironment { @module } { 0{} m } {
827   \str_set:Nx \l_stex_module_name_str { #2 }
828   \par
829   \_stex_module_args:n { #1 }
830   \_stex_module_begin_module:
831 } {
832   \_stex_module_end_module:
833 }

```

\stex_modules_heading: Code for document headers

```

834 \cs_if_exist:NTF \thesection {
835   \newcounter{module}[section]
836 }{
837   \newcounter{module}
838 }
839
840 \bool_if:NT \c_stex_showmods_bool {
841   \latexml_if:F { \RequirePackage{mdframed} }
842 }
843
844 \cs_new_protected:Nn \stex_modules_heading: {
845   \stepcounter{module}
846   \par
847   \bool_if:NT \c_stex_showmods_bool {
848     \noindent{\textbf{Module} ~

```

```

849     \cs_if_exist:NT \thesection {\thesection.}
850     \themodule ~ [\l_stex_module_name_str]
851   }
852   % TODO references
853   % \sref@label@id{Module \thesection.\themodule [\module@name]}\%
854   \str_if_empty:NTF \l_stex_module_title_str {
855     }{
856     \quad(\l_stex_module_title_str)\hfill
857   }\par
858 }
859 }

```

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

Finally:

```

860 \NewDocumentEnvironment { module } { 0 } { m } {
861   \bool_if:NT \c_stex_showmods_bool {
862     \begin{mdframed}
863   }
864   \begin{@module} [#1] {#2}
865   \stex_modules_heading:
866 }{
867   \end{@module}
868   \bool_if:NT \c_stex_showmods_bool {
869     \end{mdframed}
870   }
871 }

```

4.5.2 SMS Mode

```

872 <@@=stex_smsmode>

```

```

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

```

```

895     module,
896     @module
897   }
898 }

```

(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`

```

899 \bool_new:N \g__stex_smsmode_bool
900 \bool_set_false:N \g__stex_smsmode_bool
901 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
902   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
903 }

```

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

`_stex_smsmode_if_catcodes_p:`

`_stex_smsmode_if_catcodes:TF`

Checks whether the SMS mode category code scheme is active.

```

904 \bool_new:N \g__stex_smsmode_catcode_bool
905 \bool_set_false:N \g__stex_smsmode_catcode_bool
906 \prg_new_conditional:Nnn \_stex_smsmode_if_catcodes: { p, T, F, TF } {
907   \bool_if:NTF \g__stex_smsmode_catcode_bool
908     \prg_return_true: \prg_return_false:
909 }

```

(End definition for `_stex_smsmode_if_catcodes:TF`.)

`\stex_smsmode_set_codes:`

```

910 \cs_new_protected:Nn \stex_smsmode_set_codes: {
911   \stex_if_smsmode:T {
912     \_stex_smsmode_if_catcodes:F {
913       \bool_gset_true:N \g__stex_smsmode_catcode_bool
914       \exp_after:wN \char_gset_active_eq:NN
915         \c_backslash_str \_stex_smsmode_cs:
916       \tex_global:D \char_set_catcode_active:N \
917       \tex_global:D \char_set_catcode_other:N $
918       \tex_global:D \char_set_catcode_other:N ^
919       \tex_global:D \char_set_catcode_other:N _
920       \tex_global:D \char_set_catcode_other:N &
921       \tex_global:D \char_set_catcode_other:N ##
922     }
923   }
924 } \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.

```

925 \cs_new_protected:Nn \_stex_smsmode_unset_codes: {
926   \_stex_smsmode_if_catcodes:T {
927     \bool_gset_false:N \g__stex_smsmode_catcode_bool
928     \exp_after:wN \tex_global:D \exp_after:wN
929       \char_set_catcode_escape:N \c_backslash_str
930     \tex_global:D \char_set_catcode_math_toggle:N $
931     \tex_global:D \char_set_catcode_math_superscript:N ^
932     \tex_global:D \char_set_catcode_math_subscript:N _

```

```

933 \tex_global:D \char_set_catcode_alignment:N &
934 \tex_global:D \char_set_catcode_parameter:N ##
935 }
936 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

937 \cs_new_protected:Nn \stex_in_smsmode:nn {
938 \vbox_set:Nn \l_tmpa_box {
939 \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
940 \bool_gset_true:N \g__stex_smsmode_bool
941 \stex_smsmode_set_codes:
942 #2
943 \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
944 \stex_if_smsmode:F {
945 \_stex_smsmode_unset_codes:
946 }
947 }
948 \box_clear:N \l_tmpa_box
949 }

```

(End definition for `\stex_in_smsmode:nn`. 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:

```

950 \cs_new_protected:Nn \_stex_smsmode_cs: {
951 \str_clear:N \l_tmpa_str
952 \peek_analysis_map_inline:n {
953 % #1: token (one expansion)
954 % #2: charcode
955 % #3 catcode
956 \token_if_eq_charcode:NNTF ##3 B {
957 % token is a letter
958 \exp_args:NNo \str_put_right:Nn \l_tmpa_str { ##1 }
959 } {
960 \str_if_empty:NNTF \l_tmpa_str {
961 % we don't allow (or need) single non-letter CSs
962 % for now
963 \peek_analysis_map_break:
964 }{
965 \str_if_eq:onTF \l_tmpa_str { begin } {
966 \peek_analysis_map_break:n {
967 \exp_after:wN \_stex_smsmode_checkbegin:n ##1
968 }
969 } {
970 \str_if_eq:onTF \l_tmpa_str { end } {
971 \peek_analysis_map_break:n {
972 \exp_after:wN \_stex_smsmode_checkend:n ##1
973 }
974 } {
975 \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
976 \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
977 \g_stex_smsmode_allowedmacros_tl

```

```

978         { \use:c{\l_tmpa_str} } {
979         \stex_debug:n{Executing~1:~\l_tmpa_str}
980         \peek_analysis_map_break:n {
981             \exp_after:wN \l_tmpa_tl ##1
982         }
983     } {
984         \exp_args:NNNo \exp_args:NNo \tl_if_in:NnTF
985         \g_stex_smsmode_allowedmacros_escape_tl
986         { \use:c{\l_tmpa_str} } {
987             \stex_debug:n{Executing~2:~\l_tmpa_str}
988             % TODO \__stex_smsmode_rescan_cs:
989             % \exp_after:wN \exp_after:wN \exp_after:wN
990             % \token_if_eq_charcode:NNTF \exp_after:wN \c_backslash_str ##1 {
991             %     \peek_analysis_map_break:n {
992             %         \__stex_smsmode_unset_codes:
993             %         \__stex_smsmode_rescan_cs:
994             %     }
995             % } {
996             %     \peek_analysis_map_break:n {
997             %         \__stex_smsmode_unset_codes:
998             %         \exp_after:wN \l_tmpa_tl ##1
999             %     }
1000             % }
1001             % } {
1002             %     \peek_analysis_map_break:n { ##1 }
1003             % }
1004         }
1005     }
1006 }
1007 }
1008 }
1009 }
1010 }

```

(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:

```

1011 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1012     \str_clear:N \l_tmpb_str
1013     \peek_analysis_map_inline:n {
1014         \token_if_eq_charcode:NNTF ##3 B {
1015             % token is a letter
1016             \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1017         } {
1018             \peek_analysis_map_break:n {
1019                 \exp_after:wN \use:c \exp_after:wN {
1020                     \exp_after:wN \l_tmpa_str\exp_after:wN
1021                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1022             }
1023         }
1024     }
1025 }

```

(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.

```

1026 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1027   \str_set:Nn \l_tmpa_str { #1 }
1028   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1029     \__stex_smsmode_unset_codes:
1030     \begin{#1}
1031   }
1032 }
```

(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.

```

1033 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1034   \str_set:Nn \l_tmpa_str { #1 }
1035   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1036     \end{#1}
1037   }
1038 }
```

(End definition for `__stex_smsmode_checkend:n`.)

4.5.3 Inheritance

```

1039 <@@=stex_importmodule>
```

`\stex_import_module_uri:nn`

```

1040 \cs_new_protected:Nn \stex_import_module_uri:nn {
1041   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1042   \str_set:Nx \l__stex_importmodule_path_str { #2 }
1043   \str_if_empty:NT \l__stex_importmodule_archive_str {
1044     \prop_if_empty:NF \l_stex_current_repository_prop {
1045       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
1046     }
1047   }
1048
1049   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1050   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1051   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1052
1053   \str_if_empty:NTF \l__stex_importmodule_archive_str {
1054     \stex_modules_current_namespace:
1055     \str_if_empty:NF \l__stex_importmodule_path_str {
1056       \str_set:Nx \l_stex_module_ns_str {
1057         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1058       }
1059     }
1060   }{
1061     \stex_require_repository:n \l__stex_importmodule_archive_str
1062     \prop_get:cnN { c_stex_mathhub \l__stex_importmodule_archive_str _manifest_prop } { ns }
1063     \l_stex_module_ns_str
1064     \str_if_empty:NF \l__stex_importmodule_path_str {
1065       \str_set:Nx \l_stex_module_ns_str {
1066         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1067       }
1068     }
1069   }
```

```

1069 }
1070 }

```

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

```

\l__stex_importmodule_name_str Store the return values of \stex_import_module_uri:nn.
\l__stex_importmodule_archive_str 1071 \str_new:N \l__stex_importmodule_name_str
\l__stex_importmodule_path_str 1072 \str_new:N \l__stex_importmodule_archive_str
\l__stex_importmodule_file_str 1073 \str_new:N \l__stex_importmodule_path_str
1074 \str_new:N \g__stex_importmodule_file_str

```

(End definition for `\l__stex_importmodule_name_str` and others.)

```

\stex_import_require_module:nnnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1075 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1076   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1077     % \stex_debug:n{Arguments: #1, #2, #3, #4}
1078
1079     % archive
1080     \str_set:Nx \l_tmpa_str { #2 }
1081     \str_if_empty:NTF \l_tmpa_str {
1082       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1083     } {
1084       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1085       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1086       \seq_put_right:Nn \l_tmpa_seq { source }
1087     }
1088
1089     % path
1090     \str_set:Nx \l_tmpb_str { #3 }
1091     \str_if_empty:NTF \l_tmpb_str {
1092       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1093
1094       \cs_if_exist:NTF \language_name {
1095         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1096           { \language_name } \l_tmpb_str {
1097           \msg_set:nnn{stex}{error/unknownlanguage}{
1098             Unknown~language~\language_name
1099           }
1100           \msg_error:nn{stex}{error/unknownlanguage}
1101         }
1102       } {
1103         \str_clear:N \l_tmpb_str
1104       }
1105
1106       \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1107       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1108         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1109       }{
1110         \stex_debug:n{Checking~\l_tmpa_str.tex}
1111         \IfFileExists{ \l_tmpa_str.tex }{
1112           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1113         }{
1114           % try english as default

```

```

1115     \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1116     \IfFileExists{ \l_tmpa_str.en.tex }{
1117         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1118     }{
1119         \msg_set:nnn{stex}{error/modulemissing}{
1120             No~file~for~module~#1?#4~found
1121         }
1122         \msg_error:nn{stex}{error/modulemissing}
1123     }
1124 }
1125 }
1126
1127 } {
1128     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1129     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1130
1131     \cs_if_exist:NTF \language_name {
1132         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1133             { \language_name } \l_tmpb_str {
1134                 \msg_set:nnn{stex}{error/unknownlanguage}{
1135                     Unknown~language~\language_name
1136                 }
1137                 \msg_error:nn{stex}{error/unknownlanguage}
1138             }
1139     } {
1140         \str_clear:N \l_tmpb_str
1141     }
1142
1143     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1144
1145     \stex_debug:n{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1146     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1147         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1148     }{
1149         \stex_debug:n{Checking~\l_tmpa_str/#4.tex}
1150         \IfFileExists{ \l_tmpa_str/#4.tex }{
1151             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1152         }{
1153             % try english as default
1154             \stex_debug:n{Checking~\l_tmpa_str/#4.en.tex}
1155             \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1156                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1157             }{
1158                 \stex_debug:n{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1159                 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1160                     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1161                 }{
1162                     \stex_debug:n{Checking~\l_tmpa_str.tex}
1163                     \IfFileExists{ \l_tmpa_str.tex }{
1164                         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1165                     }{
1166                         % try english as default
1167                         \stex_debug:n{Checking~\l_tmpa_str.en.tex}
1168                         \IfFileExists{ \l_tmpa_str.en.tex }{

```

```

1169         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1170     }{
1171         \msg_set:nnn{stex}{error/modulemissing}{
1172             No~file~for~module~#1?#4~found
1173         }
1174         \msg_error:nn{stex}{error/modulemissing}
1175     }
1176 }
1177 }
1178 }
1179 }
1180 }
1181 }
1182
1183 \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1184 \seq_clear:N \g_stex_modules_in_file_seq
1185 % \exp_args:Nnx \use:nn {
1186     \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1187         \prop_clear:N \l_stex_current_module_prop
1188         \str_set:Nx \l_tmpb_str { #2 }
1189         \str_if_empty:NF \l_tmpb_str {
1190             \stex_set_current_repository:n { #2 }
1191         }
1192         \stex_debug:n{Loading~\g__stex_importmodule_file_str}
1193         \input { \g__stex_importmodule_file_str }
1194     }
1195 % }{
1196
1197 % }
1198 \prop_gput:Noo \g_stex_module_files_prop
1199 \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1200 \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1201
1202 \stex_if_module_exists:nF { #1 ? #4 } {
1203     \msg_set:nnn{stex}{error/modulemissing}{
1204         Module~#1?#4~not~found~in~file~\g__stex_importmodule_file_str
1205     }
1206     \msg_error:nn{stex}{error/modulemissing}
1207 }
1208 }
1209 % activate
1210 \stex_debug:n{Activating~module~#1?#4}
1211 \seq_put_right:Nx \l_stex_all_modules_seq {
1212     #1 ? #4
1213 }
1214 \prop_item:cn { c_stex_module_#1?#4_prop } { content }
1215 }

```

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

`\importmodule`

```

1216 \NewDocumentCommand \importmodule { 0{} m } {
1217     \stex_import_module_uri:nn { #1 } { #2 }
1218     \stex_debug:n{Importing~module:~

```

```

1219 \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1220 }
1221 \stex_if_smsmode:F {
1222   \stex_import_require_module:nnnn
1223   { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1224   { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1225   \stex_annotate_invisible:nnn
1226   {import} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1227 }
1228 \exp_args:Nx \stex_add_to_current_module:n {
1229   \stex_import_require_module:nnnn
1230   { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1231   { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1232 }
1233 \exp_args:Nx \stex_add_import_to_current_module:n {
1234   \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1235 }
1236 \stex_smsmode_set_codes:
1237 }

```

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

`\usemodule`

```

1238 \NewDocumentCommand \usemodule { 0{} m } {
1239   \stex_if_smsmode:F {
1240     \stex_import_module_uri:nn { #1 } { #2 }
1241     \stex_import_require_module:nnnn
1242     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1243     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1244     \stex_annotate_invisible:nnn
1245     {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1246   }
1247   \stex_smsmode_set_codes:
1248 }

```

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

`\g_stex_modules_in_file_seq` `\g_stex_module_files_prop`

```

1249 \seq_new:N \g_stex_modules_in_file_seq
1250 \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

```

1251 <@@=stex_symdecl>

```

`\l_stex_all_symbols_seq` Stores all available symbols

```

1252 \prop_new:N \l_stex_all_symbols_seq

```

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

`\STEXsymbol`

```
1253 \NewDocumentCommand \STEXsymbol { m } {  
1254   \stex_get_symbol:n { #1 }  
1255   \exp_args:No  
1256   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }  
1257 }
```

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

`symdecl` arguments:

```
1258 \keys_define:nn { stex / symdecl } {  
1259   name      .tl_set:x:N = \l_stex_symdecl_name_str ,  
1260   local     .bool_set:N = \l_stex_symdecl_local_bool ,  
1261   args      .tl_set:x:N = \l_stex_symdecl_args_str ,  
1262   type      .tl_set:N   = \l_stex_symdecl_type_tl ,  
1263   align     .tl_set:N   = \l_stex_symdecl_align_str , % TODO(?)  
1264   gfc       .tl_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)  
1265   specializes .tl_set:N = \l_stex_symdecl_specializes_str , % TODO(?)  
1266 }  
1267  
1268 \bool_new:N \l_stex_symdecl_make_macro_bool  
1269  
1270 \cs_new_protected:Nn \__stex_symdecl_args:n {  
1271   \str_clear:N \l_stex_symdecl_name_str  
1272   \str_clear:N \l_stex_symdecl_args_str  
1273   \bool_set_false:N \l_stex_symdecl_local_bool  
1274   \tl_clear:N \l_stex_symdecl_type_tl  
1275  
1276   \keys_set:nn { stex / symdecl } { #1 }  
1277  
1278   \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str  
1279     \l_stex_symdecl_name_str  
1280   \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str  
1281     \l_stex_symdecl_args_str  
1282 }
```

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

```
1283 \cs_new_protected:Npn \symdecl {  
1284   \peek_charcode_remove:NTF * {  
1285     \bool_set_false:N \l_stex_symdecl_make_macro_bool  
1286     \__stex_symdecl_  
1287   } {  
1288     \bool_set_true:N \l_stex_symdecl_make_macro_bool  
1289     \__stex_symdecl_  
1290   }  
1291 }  
1292  
1293 \NewDocumentCommand \__stex_symdecl_: { 0{} m } {  
1294   \__stex_symdecl_args:n { #1 }  
1295   \tl_clear:N \l_stex_symdecl_definiens_tl  
1296   \stex_symdecl_do:n { #2 }  
1297   \stex_smsmode_set_codes:  
1298 }
```

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

`\abbrdef`

```

1299 \NewDocumentCommand \abbrdef { 0{} m m } {
1300   \__stex_symdecl_args:n { #1 }
1301   \tl_set:Nn \l_stex_symdecl_definiens_tl { #3 }
1302   \bool_set_true:N \l_stex_symdecl_make_macro_bool
1303   \stex_symdecl_do:n { #2 }
1304 }

```

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

`\stex_symdecl_do:n`

```

1305 \cs_new_protected:Nn \stex_symdecl_do:n {
1306   \stex_if_in_module:F {
1307     % TODO throw error? some default namespace?
1308   }
1309
1310   \str_if_empty:NT \l_stex_symdecl_name_str {
1311     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1312   }
1313
1314   \prop_if_exist:cT { g_stex_symdecl_
1315     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1316     \prop_item:Nn \l_stex_current_module_prop {name} ?
1317     \l_stex_symdecl_name_str
1318     _prop
1319   }{
1320     % TODO throw error (beware of circular dependencies)
1321   }
1322
1323   \prop_clear:N \l_tmpa_prop
1324   \prop_put:Nnx \l_tmpa_prop { module } {
1325     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1326     \prop_item:Nn \l_stex_current_module_prop {name}
1327   }
1328   \seq_clear:N \l_tmpa_seq
1329   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1330   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1331   \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1332   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1333
1334   \exp_args:No \stex_add_constant_to_current_module:n {
1335     \l_stex_symdecl_name_str
1336   }
1337
1338   % arity/args
1339   \int_zero:N \l_tmpb_int
1340
1341   \bool_set_true:N \l_tmpa_bool
1342   \str_map_inline:Nn \l_stex_symdecl_args_str {
1343     \token_case_meaning:NnF ##1 {
1344       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1345       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```

```

1346     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1347     {\tl_to_str:n a} {
1348         \bool_set_false:N \l_tmpa_bool
1349         \int_incr:N \l_tmpb_int
1350     }
1351     {\tl_to_str:n B} {
1352         \bool_set_false:N \l_tmpa_bool
1353         \int_incr:N \l_tmpb_int
1354     }
1355 }{
1356     \msg_set:nnn{stex}{error/wrongargs}{
1357         args~value~in~symbol~declaration~for~
1358         \prop_item:Nn \l_stex_current_module_prop {ns} ?
1359         \prop_item:Nn \l_stex_current_module_prop {name} ?
1360         \l_stex_symdecl_name_str ~
1361         needs~to~be~
1362         i,~a,~b~or~B,~but~##1~given
1363     }
1364     \msg_error:nn{stex}{error/wrongargs}
1365 }
1366 }
1367 \bool_if:NTF \l_tmpa_bool {
1368     % possibly numeric
1369     \str_if_empty:NTF \l_stex_symdecl_args_str {
1370         \prop_put:Nnn \l_tmpa_prop { args } {}
1371         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1372     }{
1373         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1374         \prop_put:Nnn \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1375         \str_clear:N \l_tmpa_str
1376         \int_step_inline:nn \l_tmpa_int {
1377             \str_put_right:Nn \l_tmpa_str i
1378         }
1379         \prop_put:Nnn \l_tmpa_prop { args } { \l_tmpa_str }
1380     }
1381 } {
1382     \prop_put:Nnn \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1383     \prop_put:Nnn \l_tmpa_prop { arity }
1384     { \str_count:N \l_stex_symdecl_args_str }
1385 }
1386 \prop_put:Nnn \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1387
1388
1389 % semantic macro
1390
1391 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1392     \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1393         \prop_item:Nn \l_tmpa_prop { module } ?
1394         \prop_item:Nn \l_tmpa_prop { name }
1395     } }
1396
1397     \bool_if:NF \l_stex_symdecl_local_bool {
1398         \exp_args:Nx \stex_add_to_current_module:n {
1399             \tl_set:cx { #1 } { \stex_invoke_symbol:n {

```



```

1400         \prop_item:Nn \l_tmpa_prop { module } ?
1401         \prop_item:Nn \l_tmpa_prop { name }
1402     } }
1403 }
1404 }
1405 }
1406
1407 % add to all symbols
1408
1409 \bool_if:NF \l_stex_symdecl_local_bool {
1410     \exp_args:Nx \stex_add_to_current_module:n {
1411         \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1412             \prop_item:Nn \l_tmpa_prop { module } ?
1413             \prop_item:Nn \l_tmpa_prop { name }
1414         }
1415     }
1416 }
1417
1418 \stex_debug:n{New~symbol:~
1419     \prop_item:Nn \l_tmpa_prop { module } ?
1420     \prop_item:Nn \l_tmpa_prop { name }^^J
1421     Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1422     Args:~\prop_item:Nn \l_tmpa_prop { args }
1423 }
1424
1425 \prop_gset_eq:cn {
1426     g_stex_symdecl_
1427     \prop_item:Nn \l_tmpa_prop { module } ?
1428     \prop_item:Nn \l_tmpa_prop { name }
1429     _prop
1430 } \l_tmpa_prop
1431
1432 \stex_if_smsmode:TF {
1433     \bool_if:NF \l_stex_symdecl_local_bool {
1434         \exp_args:Nx \stex_addtosms:n {
1435             \prop_gset_from_keyval:cn {
1436                 g_stex_symdecl_
1437                 \prop_item:Nn \l_tmpa_prop { module } ?
1438                 \prop_item:Nn \l_tmpa_prop { name }
1439                 _prop
1440             } {
1441                 name      = \prop_item:Nn \l_tmpa_prop { name }
1442                 module    = \prop_item:Nn \l_tmpa_prop { module }
1443                 notations = \prop_item:Nn \l_tmpa_prop { notations }
1444                 local     = \prop_item:Nn \l_tmpa_prop { local }
1445                 type      = \prop_item:Nn \l_tmpa_prop { type }
1446                 args      = \prop_item:Nn \l_tmpa_prop { args }
1447                 arity     = \prop_item:Nn \l_tmpa_prop { arity }
1448                 assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1449             }
1450             \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1451                 \prop_item:Nn \l_tmpa_prop { module } ?
1452                 \prop_item:Nn \l_tmpa_prop { name }
1453             }

```

```

1454     }
1455   }
1456   ){
1457     \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1458       \prop_item:Nn \l_tmpa_prop { module } ?
1459       \prop_item:Nn \l_tmpa_prop { name }
1460     }
1461     \stex_annotate_invisible:nnn {symdecl} {
1462       \prop_item:Nn \l_tmpa_prop { module } ?
1463       \prop_item:Nn \l_tmpa_prop { name }
1464     } {
1465       \stex_annotate_invisible:nnn{type}{}{${\l_stex_symdecl_type_tl$}
1466       \stex_annotate_invisible:nnn{args}{}{
1467         \prop_item:Nn \l_tmpa_prop { args }
1468       }
1469       \stex_annotate_invisible:nnn{macroname}{}{#1}
1470       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1471         \stex_annotate_invisible:nnn{definiens}{}{
1472           ${\l_stex_symdecl_definiens_tl$}
1473         }
1474       }
1475     }
1476   }

```

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

`\stex_get_symbol:n`

```

1477 \str_new:N \l_stex_get_symbol_uri_str
1478
1479 \cs_new_protected:Nn \stex_get_symbol:n {
1480   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1481     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1482   }{
1483     % argument is a string
1484     % is it a command name?
1485     \cs_if_exist:cTF { #1 }{
1486       \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1487     }{
1488       % argument is not a command name
1489       \prop_get:NnN \l_stex_current_module_prop
1490         { constants } \l_tmpa_seq
1491       \seq_if_in:NnTF \l_tmpa_seq { #1 } {
1492         \str_set:Nx \l_stex_get_symbol_uri_str {
1493           \prop_item:Nn \l_stex_current_module_prop { ns } ?
1494           \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1495         }
1496       } {
1497         \tl_set:Nn \l_tmpa_tl {
1498           \msg_set:nnn{stex}{error/unknownsymbol}{
1499             No~symbol~#1~found!
1500           }
1501           \msg_error:nn{stex}{error/unknownsymbol}
1502         }
1503         \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }

```

```

1504     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1505     \seq_map_inline:Nn \l_stex_all_symbols_seq {
1506         \str_set:Nn \l_tmpb_str { ##1 }
1507         \str_if_eq:eeT { \l_tmpa_str } {
1508             \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1509         } {
1510             \seq_map_break:n {
1511                 \tl_set:Nn \l_tmpa_tl {
1512                     \str_set:Nn \l_stex_get_symbol_uri_str {
1513                         ##1
1514                     }
1515                 }
1516             }
1517         }
1518     }
1519     \l_tmpa_tl
1520 }
1521 % \l_stex_all_symbols_seq
1522 }
1523 }
1524 }
1525
1526 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1527     \tl_set:Nx \l_tmpa_tl { #1 }
1528     \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
1529         \stex_invoke_symbol:n {
1530             \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1531                 { \tl_tail:N \l_tmpa_tl }
1532             \tl_if_single:NTF \l_tmpa_tl {
1533                 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1534                     \exp_after:wN \str_set:Nn \exp_after:wN
1535                         \l_stex_get_symbol_uri_str \l_tmpa_tl
1536                 }{
1537                     % TODO
1538                     % tail is not a single group
1539                 }
1540             }{
1541                 % TODO
1542                 % tail is not a single group
1543             }
1544         }{
1545             % TODO
1546             % head is not \stex_invoke_symbol:n
1547         }
1548     }

```

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

4.7 Notations

```

1549 <@@=stex_notation>
      notation arguments:
1550 \keys_define:nn { stex / notation } {
1551     lang      .tl_set_x:N = \l__stex_notation_lang_str ,

```

```

1552   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1553   prec    .tl_set_x:N = \l__stex_notation_prec_str ,
1554   unknown .code:n      = \str_set:Nx
1555       \l__stex_notation_variant_str \l_keys_key_str
1556 }
1557
1558 \cs_new_protected:Nn \__stex_notation_args:n {
1559   \str_clear:N \l__stex_notation_lang_str
1560   \str_clear:N \l__stex_notation_variant_str
1561   \str_clear:N \l__stex_notation_prec_str
1562
1563   \keys_set:nn { stex / notation } { #1 }
1564
1565   \str_set:Nx \l__stex_notation_lang_str \l__stex_notation_lang_str
1566   \str_set:Nx \l__stex_notation_variant_str \l__stex_notation_variant_str
1567   \str_set:Nx \l__stex_notation_prec_str \l__stex_notation_prec_str
1568 }

```

\notation

```

1569 \NewDocumentCommand \notation { 0{} m } {
1570   \__stex_notation_args:n { #1 }
1571   \tl_clear:N \l_stex_symdecl_definiens_tl
1572   \stex_get_symbol:n { #2 }
1573   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1574 }

```

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

\stex_notation_do:nn

```

1575 \cs_new_protected:Nn \stex_notation_do:nn {
1576   \prop_set_eq:Nc \l_tmpa_prop {
1577     g_stex_symdecl_ #1 _prop
1578   }
1579
1580   \prop_clear:N \l_tmpb_prop
1581   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1582   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1583   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1584
1585   % precedences
1586   \seq_clear:N \l_tmpb_seq
1587   \exp_args:NNno
1588   \str_if_empty:NTF \l__stex_notation_prec_str {
1589     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1590     \int_compare:nNnTF \l_tmpa_str = 0 {
1591       \exp_args:NNnx
1592       \prop_put:Nno \l_tmpb_prop { opprec }
1593       { \infprec }
1594     }{
1595       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1596     }
1597   } {
1598     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
1599     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1600       \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str

```

```

1601 \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
1602 \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
1603 \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
1604 \seq_map_inline:Nn \l_tmpa_seq {
1605 \seq_put_right:Nn \l_tmpb_seq { ##1 }
1606 }
1607 }
1608 \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1609 }{
1610 \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1611 \int_compare:nNnTF \l_tmpa_str = 0 {
1612 \exp_args:NNnx
1613 \prop_put:Nno \l_tmpb_prop { opprec }
1614 { \infprec }
1615 }{
1616 \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1617 }
1618 }
1619 }
1620
1621 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1622 \int_step_inline:nn { \l_tmpa_str } {
1623 \seq_pop_left:NNT \l_tmpa_seq \l_tmpb_str {
1624 \exp_args:NNx
1625 \seq_put_right:Nn \l_tmpb_seq {
1626 \prop_item:Nn \l_tmpb_prop { opprec }
1627 }
1628 }
1629 }
1630
1631 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
1632 \tl_clear:N \l_tmpa_tl
1633
1634 \int_compare:nNnTF \l_tmpa_str = 0 {
1635 \exp_args:NNe
1636 \cs_set:Npn \l__stex_notation_macrocode_cs {
1637 \stex_term_math_oms: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 { \exp_not:n { #2 } }
1641 }
1642 \__stex_notation_final:
1643 }{
1644 \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1645 \str_if_in:NnTF \l_tmpb_str b {
1646 \exp_args:Nne \use:nn
1647 {
1648 \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1649 \cs_set:Npn \l_tmpa_str { {
1650 \stex_term_math_omb:nnnn { #1 }
1651 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1652 { \prop_item:Nn \l_tmpb_prop { opprec } }
1653 { \exp_not:n { #2 } }
1654 }}

```

```

1655   }{
1656     \str_if_in:NnTF \l_tmpb_str B {
1657       \exp_args:Nne \use:nn
1658       {
1659         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1660         \cs_set:Npn \l_tmpa_str } { {
1661           \stex_term_math_omb:nnnn { #1 }
1662           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1663           { \prop_item:Nn \l_tmpb_prop { opprec } }
1664           { \exp_not:n { #2 } }
1665         } }
1666       }{
1667         \exp_args:Nne \use:nn
1668         {
1669           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1670           \cs_set:Npn \l_tmpa_str } { {
1671             \stex_term_math_oma:nnnn { #1 }
1672             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1673             { \prop_item:Nn \l_tmpb_prop { opprec } }
1674             { \exp_not:n { #2 } }
1675           } }
1676         }
1677       }
1678
1679       \int_zero:N \l_tmpa_int
1680       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1681       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1682       \__stex_notation_arguments:
1683     }
1684   }

```

(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

```

1685 \cs_new_protected:Nn \__stex_notation_arguments: {
1686   \int_incr:N \l_tmpa_int
1687   \str_if_empty:NnTF \l_tmpa_str {
1688     \__stex_notation_final:
1689   }{
1690     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1691     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1692     \str_if_eq:NnTF \l_tmpb_str a {
1693       \__stex_notation_argument_assoc:n
1694     }{
1695       \str_if_eq:NnTF \l_tmpb_str B {
1696         \__stex_notation_argument_assoc:n
1697       }{
1698         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1699         \tl_put_right:Nx \l_tmpa_tl {
1700           { \stex_term_math_arg:nnn
1701             { \int_use:N \l_tmpa_int }
1702             { \l_tmpb_str }
1703             { ####\int_use:N \l_tmpa_int }
1704           }

```

```

1705     }
1706     \__stex_notation_arguments:
1707   }
1708 }
1709 }
1710 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

1711 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
1712   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1713   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
1714   \tl_put_right:Nx \l_tmpa_tl {
1715     { \stex_term_math_assoc_arg:nnnn
1716       { \int_use:N \l_tmpa_int }
1717       { \l_tmpb_str }
1718       { \l_tmpa_cs {#####1} {#####2} }
1719       { ####\int_use:N \l_tmpa_int }
1720     }
1721   }
1722   \__stex_notation_arguments:
1723 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

1724 \cs_new_protected:Nn \__stex_notation_final: {
1725   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
1726   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
1727   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1728   \exp_args:Nne \use:nn
1729   {
1730     \cs_generate_from_arg_count:cNnn {
1731       stex_notation_ \l_tmpa_str \c_hash_str
1732       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1733       _cs
1734     }
1735     \cs_set:Npn \l_tmpb_str } { {
1736       \exp_after:wN \exp_after:wN \exp_after:wN
1737       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
1738       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
1739     } }
1740
1741   \stex_debug:n{
1742     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1743     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
1744     Operator~precedence:~
1745     \prop_item:Nn \l_tmpb_prop { opprec }^^J
1746     Argument~precedences:~
1747     \seq_use:Nn \l_tmpa_seq { ,~ }^^J
1748     Notation: \cs_meaning:c {
1749       stex_notation_ \l_tmpa_str \c_hash_str
1750       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str

```

```

1751     _cs
1752   }
1753 }
1754
1755 \prop_gset_eq:cN {
1756   g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1757   \c_hash_str \l__stex_notation_lang_str _prop
1758 } \l_tmpb_prop
1759
1760 \exp_args:Nx
1761 \stex_add_to_current_module:n {
1762   \prop_get:cnN {
1763     g_stex_symdecl_
1764     \prop_item:Nn \l_tmpb_prop { symbol }
1765     _prop
1766   } { notations } \exp_not:N \l_tmpa_seq
1767   \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
1768     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1769   }
1770   \prop_put:cno {
1771     g_stex_symdecl_
1772     \prop_item:Nn \l_tmpb_prop { symbol }
1773     _prop
1774   } { notations } \exp_not:N \l_tmpa_seq
1775 }
1776
1777 \stex_if_smsmode:TF {
1778   \stex_smsmode_set_codes:
1779   \exp_args:Nx \stex_addtosms:n {
1780     \prop_gset_from_keyval:cn {
1781       g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
1782       \c_hash_str \l__stex_notation_lang_str _prop
1783     } {
1784       symbol      = \prop_item:Nn \l_tmpb_prop { symbol }
1785       language    = \prop_item:Nn \l_tmpb_prop { language }
1786       variant     = \prop_item:Nn \l_tmpb_prop { variant }
1787       opprec      = \prop_item:Nn \l_tmpb_prop { opprec }
1788       argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }
1789     }
1790   }
1791 }{
1792   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
1793   \seq_put_right:Nx \l_tmpa_seq {
1794     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1795   }
1796   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1797   \prop_set_eq:cN {
1798     g_stex_symdecl_ \l_tmpa_str _prop
1799   } \l_tmpa_prop
1800
1801   % HTML annotations
1802   \stex_annotate_invisible:nnn { notation }
1803   { \prop_item:Nn \l_tmpb_prop { symbol } } {
1804     \stex_annotate_invisible:nnn { notationfragment }

```



```

1805     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
1806 \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
1807 \stex_annotate_invisible:nnn { precedence }
1808   { \prop_item:Nn \l_tmpb_prop { opprec };
1809     \seq_use:Nn \l_tmpa_seq { x }
1810   }{}
1811
1812 \int_zero:N \l_tmpa_int
1813 \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1814 \tl_clear:N \l_tmpa_tl
1815 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
1816   \int_incr:N \l_tmpa_int
1817   \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
1818   \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
1819   \str_if_eq:VnTF \l_tmpb_str a {
1820     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1821       \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1822       \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
1823     } }
1824   }{
1825     \str_if_eq:VnTF \l_tmpb_str B {
1826       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1827         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
1828         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
1829       } }
1830     }{
1831       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
1832         \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
1833       } }
1834     }
1835   }
1836 }
1837 \stex_annotate_invisible:nnn { notationcomp }{}{
1838   $ \exp_args:Nno \use:nn { \use:c {
1839     stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }
1840     \c_hash_str \l__stex_notation_variant_str
1841     \c_hash_str \l__stex_notation_lang_str _cs
1842   } } { \l_tmpa_tl } $
1843 }
1844 }
1845 }
1846 }

```

(End definition for _stex_notation_final:.)

\symdef

```

1847 \keys_define:nn { stex / symdef } {
1848   name .tl_set_x:N = \l_stex_symdecl_name_str ,
1849   local .bool_set:N = \l_stex_symdecl_local_bool ,
1850   args .tl_set_x:N = \l_stex_symdecl_args_str ,
1851   type .tl_set:N = \l_stex_symdecl_type_tl ,
1852   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1853   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1854   prec .tl_set_x:N = \l__stex_notation_prec_str ,

```

```

1855   unknown .code:n      = \str_set:Nx
1856       \l__stex_notation_variant_str \l_keys_key_str
1857 }
1858
1859 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
1860   \str_clear:N \l_stex_symdecl_name_str
1861   \str_clear:N \l_stex_symdecl_args_str
1862   \bool_set_false:N \l_stex_symdecl_local_bool
1863   \tl_clear:N \l_stex_symdecl_type_tl
1864   \str_clear:N \l__stex_notation_lang_str
1865   \str_clear:N \l__stex_notation_variant_str
1866   \str_clear:N \l__stex_notation_prec_str
1867
1868   \keys_set:nn { stex /symdef } { #1 }
1869
1870   \exp_args:NNo \str_set:Nn \l_stex_symdecl_name_str
1871     \l_stex_symdecl_name_str
1872   \exp_args:NNo \str_set:Nn \l_stex_symdecl_args_str
1873     \l_stex_symdecl_args_str
1874   \exp_args:NNo \str_set:Nn \l__stex_notation_lang_str
1875     \l__stex_notation_lang_str
1876   \exp_args:NNo \str_set:Nn \l__stex_notation_variant_str
1877     \l__stex_notation_variant_str
1878   \exp_args:NNo \str_set:Nn \l__stex_notation_prec_str
1879     \l__stex_notation_prec_str
1880 }
1881
1882 \NewDocumentCommand \symdef { O{} m } {
1883   \__stex_notation_symdef_args:n { #1 }
1884   \tl_clear:N \l_stex_symdecl_definiens_tl
1885   \bool_set_true:N \l_stex_symdecl_make_macro_bool
1886   \stex_symdecl_do:n { #2 }
1887   \exp_args:Nx \stex_notation_do:nn {
1888     \prop_item:Nn \l_tmpa_prop { module } ?
1889     \prop_item:Nn \l_tmpa_prop { name }
1890   }
1891 }

```

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

`\stex_invoke_symbol:n` Invokes a semantic macro

```

1892 \cs_new_protected:Nn \stex_invoke_symbol:n {
1893   \peek_charcode_remove:NTF ! {
1894     \stex_term_custom:nn { #1 } { }
1895   } {
1896     \if_mode_math:
1897       \exp_after:wN \__stex_notation_invoke_math:n
1898     \else:
1899       \exp_after:wN \__stex_notation_invoke_text:n
1900     \fi: { #1 }
1901   }
1902 }

```

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

_stex_notation_invoke_math:n

```

1903 \cs_new_protected:Nn \__stex_notation_invoke_math:n {
1904   \peek_charcode_remove:NTF * {
1905     \__stex_notation_invoke_text:n { #1 }
1906   }{
1907     \peek_charcode:NTF [ {
1908       \__stex_notation_invoke_math:nw { #1 }
1909     }{
1910       \__stex_notation_invoke_math:nw { #1 } []
1911     }
1912   }
1913 }

```

(End definition for __stex_notation_invoke_math:n.)

_stex_notation_invoke_math:nw

```

1914 \cs_new_protected:Npn \__stex_notation_invoke_math:nw #1 [#2] {
1915   \__stex_notation_args:n { #2 }
1916   \prop_set_eq:Nc \l_tmpa_prop {
1917     g_stex_symdecl_ #1 _prop
1918   }
1919   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
1920   \seq_if_empty:NTF \l_tmpa_seq {
1921     \msg_set:nnn{stex}{error/nonotations}{
1922       Symbol~#1~used,~but~has~no~notations!
1923     }
1924     \msg_error:nn{stex}{error/nonotations}
1925   } {
1926     \seq_if_in:NxTF \l_tmpa_seq
1927       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
1928       \use:c{
1929         stex_notation_ #1 \c_hash_str
1930         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1931         _cs
1932       }
1933     }{
1934       \str_if_empty:NTF \l__stex_notation_variant_str {
1935         \str_if_empty:NTF \l__stex_notation_lang_str {
1936           \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
1937           \use:c{
1938             stex_notation_ #1 \c_hash_str \l_tmpa_str
1939             _cs
1940           }
1941         }{
1942           \msg_set:nnn{stex}{error/wrongnotation}{
1943             Symbol~#1~has~no~notation~
1944             \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
1945           }
1946           \msg_error:nn{stex}{error/wrongnotation}
1947         }
1948       }{
1949         \msg_set:nnn{stex}{error/wrongnotation}{
1950           Symbol~#1~has~no~notation~
1951           \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str

```

```

1952     }
1953     \msg_error:nn{stex}{error/wrongnotation}
1954   }
1955 }
1956 }
1957 }

```

(End definition for `__stex_notation_invoke_math:nw`.)

`_stex_notation_invoke_text:n`

```

1958 \cs_new_protected:Nn \__stex_notation_invoke_text:n {
1959   \prop_set_eq:Nc \l_tmpa_prop {
1960     g_stex_symdecl_ #1 _prop
1961   }
1962   \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
1963   \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
1964 }

```

(End definition for `_stex_notation_invoke_text:n`.)

4.8 Terms

1965 `<@@=stex_term>`

Precedences:

```

\infprec
\neginfprec
1966 \tl_const:Nx \infprec {\int_use:N \c_max_int}
1967 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
1968 \int_new:N \l__stex_term_downprec
1969 \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
1970 \tl_set:Nn \l__stex_term_left_bracket_str (
1971 \tl_set:Nn \l__stex_term_right_bracket_str )
1972 \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

```

1973 \cs_new_protected:Nn \_stex_term_maybe_brackets:nn {
1974   \int_compare:nNnTF { #1 } < \l__stex_term_downprec {
1975     \dobrackets { #2 }
1976   }{ #2 }
1977 }

```

(End definition for `_stex_term_maybe_brackets:nn`.)

\dobrackets

```
1978 \cs_new_protected:Npn \dobrackets #1 {
1979   \ThisStyle{\if D\m@switch
1980     \exp_args:Nnx \use:nn
1981     { \left\l__stex_term_left_bracket_str #1 }
1982     { \right\l__stex_term_right_bracket_str }
1983   \else
1984     \exp_args:Nnx \use:nn
1985     { \l__stex_term_left_bracket_str #1 }
1986     { \l__stex_term_right_bracket_str }
1987   \fi}
1988 }
```

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

\withbrackets

```
1989 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
1990   \exp_args:Nnx \use:nn
1991   {
1992     \tl_set:Nx \l__stex_term_left_bracket_str { #1 }
1993     \tl_set:Nx \l__stex_term_right_bracket_str { #2 }
1994     #3
1995   }
1996   {
1997     \tl_set:Nn \exp_not:N \l__stex_term_left_bracket_str
1998     {\l__stex_term_left_bracket_str}
1999     \tl_set:Nn \exp_not:N \l__stex_term_right_bracket_str
2000     {\l__stex_term_right_bracket_str}
2001   }
2002 }
```

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

\STEXinvisible

```
2003 \cs_new_protected:Npn \STEXinvisible #1 {
2004   \stex_annotate_invisible:n { #1 }
2005 }
```

(End definition for \STEXinvisible. This function is documented on page 24.)

OMDOC terms:

_stex_term_math_oms:nnnn

```
2006 \cs_new_protected:Nn \_stex_term_oms:nnn {
2007   \stex_annotate:nnn{ OMID }{ #2 }{
2008     \stex_highlight_term:nn { #1 } { #3 }
2009   }
2010 }
2011
2012 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2013   \_stex_term_maybe_brackets:nn { #3 }{
2014     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2015   }
2016 }
```

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

`_stex_term_math_oma:nnnn`

```
2017 \cs_new_protected:Nn \_stex_term_oma:nnn {
2018   \stex_annotate:nnn{ OMA }{ #2 }{
2019     \stex_highlight_term:nn { #1 } { #3 }
2020   }
2021 }
2022
2023 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2024   \__stex_term_maybe_brackets:nn { #3 }{
2025     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2026   }
2027 }
```

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

`_stex_term_math_omb:nnnn`

```
2028 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2029   \stex_annotate:nnn{ OMBIND }{ #2 }{
2030     \stex_highlight_term:nn { #1 } { #3 }
2031   }
2032 }
2033
2034 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2035   \__stex_term_maybe_brackets:nn { #3 }{
2036     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2037   }
2038 }
```

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

`_stex_term_math_arg:nnn`

```
2039 \cs_new_protected:Nn \_stex_term_arg:nn {
2040   \stex_unhighlight_term:n {
2041     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2042   }
2043 }
2044 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2045   \exp_args:Nnx \use:nn
2046     { \int_set:Nn \l__stex_term_downprec { #2 }
2047       \_stex_term_arg:nn { #1 } { #3 }
2048     }
2049     { \int_set:Nn \exp_not:N \l__stex_term_downprec { \int_use:N \l__stex_term_downprec } }
2050 }
```

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

`_stex_term_math_assoc_arg:nnnn`

```
2051 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2052   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
2053   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
2054     \tl_set:Nn \l_tmpa_tl { #4 }
2055   }{
2056     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2057     \seq_reverse:N \l_tmpa_seq

```

```

2058 \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
2059 \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
2060 \seq_map_inline:Nn \l_tmpa_seq {
2061   \tl_set:Nx \l_tmpa_tl {
2062     \exp_args:Nno
2063     \l_tmpa_cs { ##1 } { \l_tmpa_tl }
2064   }
2065 }
2066 }
2067 \exp_args:Nnno
2068 \stex_term_math_arg:nnn{#1}{#2}{ \l_tmpa_tl }
2069 }

```

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

`\stex_term_custom:nn`

```

2070 \cs_new_protected:Nn \stex_term_custom:nn {
2071   \str_set:Nn \l__stex_term_custom_uri { #1 }
2072   \str_set:Nn \l_tmpa_str { #2 }
2073   \tl_clear:N \l_tmpa_tl
2074   \int_zero:N \l_tmpa_int
2075   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2076   \__stex_term_custom_loop:
2077 }

```

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

`__stex_term_custom_loop:`

```

2078 \cs_new_protected:Nn \__stex_term_custom_loop: {
2079   \bool_set_false:N \l_tmpa_bool
2080   \bool_while_do:nn {
2081     \str_if_eq_p:ee X {
2082       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2083     }
2084   }{
2085     \int_incr:N \l_tmpa_int
2086   }
2087
2088   \peek_charcode:NTF [ {
2089     % notation/text component
2090     \__stex_term_custom_component:w
2091   } {
2092     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2093       % all arguments read => finish
2094       \__stex_term_custom_final:
2095     } {
2096       % arguments missing
2097       \peek_charcode_remove:NTF * {
2098         % invisible, specific argument position or both
2099         \peek_charcode:NTF [ {
2100           % visible specific argument position
2101           \__stex_term_custom_arg:wn
2102         } {
2103           % invisible
2104           \peek_charcode_remove:NTF * {

```

```

2105         % invisible specific argument position
2106         \__stex_term_custom_arg_inv:wn
2107     } {
2108         % invisible next argument
2109         \__stex_term_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2110     }
2111 }
2112 } {
2113     % next normal argument
2114     \__stex_term_custom_arg:wn [ \l_tmpa_int + 1 ]
2115 }
2116 }
2117 }
2118 }

```

(End definition for __stex_term_custom_loop:.)

_stex_term_custom_arg_inv:wn

```

2119 \cs_new_protected:Npn \__stex_term_custom_arg_inv:wn [ #1 ] #2 {
2120     \bool_set_true:N \l_tmpa_bool
2121     \__stex_term_custom_arg:wn [ #1 ] { #2 }
2122 }

```

(End definition for __stex_term_custom_arg_inv:wn.)

__stex_term_custom_arg:wn

```

2123 \cs_new_protected:Npn \__stex_term_custom_arg:wn [ #1 ] #2 {
2124     \str_set:Nx \l_tmpb_str {
2125         \str_item:Nn \l_tmpa_str { #1 }
2126     }
2127     \str_case:VnTF \l_tmpb_str {
2128         { X } { } % TODO throw error
2129         { i } { \__stex_term_custom_set_X:n { #1 } }
2130         { b } { \__stex_term_custom_set_X:n { #1 } }
2131         { a } { } % TODO ?
2132     }{}{
2133         % TODO throw error
2134     }
2135
2136     \bool_if:nTF \l_tmpa_bool {
2137         \tl_put_right:Nx \l_tmpa_tl {
2138             \stex_annotate_invisible:n {
2139                 \stex_term_arg:nn { \int_eval:n { #1 } }
2140                 \exp_not:n { { #2 } }
2141             }
2142         }
2143     } {
2144         \tl_put_right:Nx \l_tmpa_tl {
2145             \stex_term_arg:nn { \int_eval:n { #1 } }
2146             \exp_not:n { { #2 } }
2147         }
2148     }
2149
2150     \__stex_term_custom_loop:
2151 }

```


(End definition for _stex_term_custom_arg:wn.)

_stex_term_custom_set_X:n

```

2152 \cs_new_protected:Nn \_stex_term_custom_set_X:n {
2153   \str_set:Nx \l_tmpa_str {
2154     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2155     X
2156     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2157   }
2158 }

```

(End definition for _stex_term_custom_set_X:n.)

_stex_term_custom_component:

```

2159 \cs_new_protected:Npn \_stex_term_custom_component:w [ #1 ] {
2160   \tl_put_right:Nn \l_tmpa_tl { #1 }
2161   \_stex_term_custom_loop:
2162 }

```

(End definition for _stex_term_custom_component:.)

_stex_term_custom_final:

```

2163 \cs_new_protected:Nn \_stex_term_custom_final: {
2164   \int_compare:nNnTF \l_tmpb_int = 0 {
2165     \exp_args:Nnno \_stex_term_oms:nnn
2166   }{
2167     \str_if_in:NnTF \l_tmpa_str {b} {
2168       \exp_args:Nnno \_stex_term_ombind:nnn
2169     } {
2170       \exp_args:Nnno \_stex_term_oma:nnn
2171     }
2172   }
2173   { \l__stex_term_custom_uri } { \l__stex_term_custom_uri } { \l_tmpa_tl }
2174 }

```

(End definition for _stex_term_custom_final:.)

\stex_highlight_term:nn

```

2175 \latexml_if:F {
2176   \scalatex_if:F{
2177     \RequirePackage{pdfcomment}
2178   }
2179 }
2180
2181 \str_new:N \l__stex_term_highlight_uri_str
2182 \cs_new_protected:Nn \stex_highlight_term:nn {
2183   \latexml_if:TF {
2184     #2
2185   } {
2186     \scalatex_if:TF {
2187       #2
2188     } {
2189       \exp_args:Nnx
2190       \use:nn {
2191         \str_set:Nx \l__stex_term_highlight_uri_str { #1 }

```

```

2192         #2
2193     } {
2194         \str_set:Nx \exp_not:N \l__stex_term_highlight_uri_str
2195         { \l__stex_term_highlight_uri_str }
2196     }
2197 }
2198 }
2199 }
2200
2201 \cs_new_protected:Nn \stex_unhighlight_term:n {
2202 % \latexml_if:TF {
2203 %     #1
2204 % } {
2205 %     \scalatex_if:TF {
2206 %         #1
2207 %     } {
2208         #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2209 %     }
2210 % }
2211 }

```

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

```

\comp
\@comp
2212 \cs_new_protected:Npn \comp #1 {
2213     \str_if_empty:NF \l__stex_term_highlight_uri_str {
2214         \exp_args:Nnx \@comp { #1 } { \l__stex_term_highlight_uri_str }
2215     }
2216 }
2217
2218 \cs_new_protected:Npn \@comp #1 #2 {
2219     \pdftooltip {
2220         \textcolor{blue}{#1}
2221     } { #2 }
2222 }

```

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

\ellipses

```

2223 \cs_new_protected:Npn \ellipses {
2224     \ldots
2225 }

```

(End definition for `\ellipses`. This function is documented on page 25.)

```

2226 \ifpackageloaded{tikzinput}{
2227     \RequirePackage{stex-tikzinput}
2228 }{}
2229 \</package>

```

4.9 Auxiliary Packages

4.9.1 tikzinput

```
2230 <*tikzinput>
2231 <@@=tikzinput>
2232 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
2233 \RequirePackage{l3keys2e}
2234
2235 \keys_define:nn { tikzinput } {
2236   image .bool_set:N = \c_tikzinput_image_bool
2237 }
2238
2239 \ProcessKeysOptions { tikzinput }
2240
2241 \bool_if:NTF \c_tikzinput_image_bool {
2242   \RequirePackage{graphicx}
2243
2244   \providecommand\usetikzlibrary[]{}
2245   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
2246 }{
2247   \RequirePackage{tikz}
2248   \RequirePackage{standalone}
2249
2250   \newcommand \tikzinput [2] [] {
2251     \setkeys{Gin}{#1}
2252     \ifx \Gin@width \Gin@exclamation
2253       \ifx \Gin@height \Gin@exclamation
2254         \input { #2 }
2255       \else
2256         \resizebox{!}{ \Gin@height }{
2257           \input { #2 }
2258         }
2259       \fi
2260     \else
2261       \ifx \Gin@height \Gin@exclamation
2262         \resizebox{ \Gin@width }{!}{
2263           \input { #2 }
2264         }
2265       \else
2266         \resizebox{ \Gin@width }{ \Gin@height }{
2267           \input { #2 }
2268         }
2269       \fi
2270     \fi
2271   }
2272 }
2273
2274 \newcommand \ctikzinput [2] [] {
2275   \begin{center}
2276     \tikzinput [#1] {#2}
2277   \end{center}
2278 }
2279
2280 \@ifpackageloaded{stex}{
```

```

2281 \RequirePackage{stex-tikzinput}
2282 }{}
2283 </tikzinput>
2284 <*stex-tikzinput>
2285 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
2286 \RequirePackage{stex}
2287 \RequirePackage{tikzinput}
2288
2289 % TODO
2290
2291 </stex-tikzinput>

```

4.9.2 sTeX1 Compatibility

```

2292 <*smglom>
2293 \RequirePackage{expl3,l3keys2e}
2294 \ProvidesExplClass{smglom}{2021/08/01}{1.9}{sTeX1 compatibility}
2295 \LoadClass[border=1px,varwidth]{standalone}
2296 \setlength\textwidth{15cm}
2297 %\g@addto@macro{\@parboxrestore}{\setlength\parskip{\baselineskip}}
2298 \DeclareOption{mh}{}
2299 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
2300 \ProcessOptions
2301
2302 \RequirePackage{stex-compatibility}
2303 </smglom>
2304
2305 <*compat>
2306 <@@=stex_deprec>
2307 \ProvidesExplPackage{stex-compatibility}{2021/08/01}{1.9}{bla}
2308 \RequirePackage[debug,lang={de,en}]{stex}
2309
2310 \NewDocumentEnvironment { mhmodnl } { 0{} m m } {
2311   \msg_set:nnn{stex}{warning/deprecated}{
2312     \\\
2313     Environment~mhmodnl~is~deprected! \\\
2314     Please~update~module~#2~in~file~
2315     \stex_path_to_string:N \g_stex_currentfile_seq!
2316     \\\ \\\
2317   }
2318   \msg_warning:nn{stex}{warning/deprecated}
2319
2320   \begin{module}[#1,lang=#3]{#2}
2321     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
2322     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2323     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
2324     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
2325     \input { \stex_path_to_string:N \l_tmpa_seq / \l_tmpa_str.tex }
2326   } {
2327     \end{module}
2328   }
2329
2330 \NewDocumentEnvironment { modsig } { 0{} m } {
2331   \stex_if_in_module:TF {
2332     \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str

```

```

2333 \str_set:Nn \l_tmpb_str { #2 }
2334 \str_if_eq:NNTF \l_tmpa_str \l_tmpb_str {
2335   \prop_set_eq:NN \l_modsig_old_module_prop \l_stex_current_module_prop
2336   \begin{@module}{modsig-#2}
2337   % \prop_set_eq:NN \l_stex_current_module_prop \l_modsig_old_module_prop
2338 } {
2339   \begin{@module}{#2}
2340 }
2341 } {
2342   \begin{@module}{#2}
2343 }
2344 }{
2345   \end{@module}
2346   \AddToHookNext { env / modsig / after }{
2347     \stex_if_in_module:T {
2348       \prop_get:NnN \l_stex_current_module_prop {name} \l_tmpa_str
2349       \str_set:Nn \l_tmpb_str { #2 }
2350       \str_if_eq:NNT \l_tmpa_str \l_tmpb_str {
2351         % \xdef \g_stex_module_after_group_tl {
2352           \stex_if_smsmode:TF {
2353             \exp_args:Nx
2354             \stex_add_to_current_module:n {
2355               \stex_debug:n{Activating~signature~of~#2}
2356               \exp_not:N \prop_item:cn { c_stex_module_
2357               \prop_item:Nn \l_stex_current_module_prop {ns} ?
2358               \prop_item:Nn \l_stex_current_module_prop {name}
2359               / modsig-#2_prop } { content }
2360             }
2361           }
2362         {
2363           \gdef \g_stex_module_after_group_tl {
2364             \stex_debug:n{Activating~signature~of~#2}
2365
2366             \seq_put_right:Nx \l_stex_all_modules_seq {
2367               \prop_item:Nn \l_stex_current_module_prop {ns} ?
2368               \prop_item:Nn \l_stex_current_module_prop {name}
2369               / modsig-#2_prop
2370             }
2371             \prop_item:cn { c_stex_module_
2372             \prop_item:Nn \l_stex_current_module_prop {ns} ?
2373             \prop_item:Nn \l_stex_current_module_prop {name}
2374             / modsig-#2_prop } { content }
2375             \exp_args:Nx
2376             \stex_add_to_current_module:n {
2377               \stex_debug:n{Activating~signature~of~#2}
2378               \exp_not:N \prop_item:cn { c_stex_module_
2379               \prop_item:Nn \l_stex_current_module_prop {ns} ?
2380               \prop_item:Nn \l_stex_current_module_prop {name}
2381               / modsig-#2_prop } { content }
2382             }
2383           }
2384           \aftergroup \g_stex_module_after_group_tl
2385         }
2386         %

```

```

2387 %      \aftergroup \g_stex_module_after_group_tl
2388     }
2389   }
2390 }
2391 }
2392
2393 \cs_new_protected:Npn \gimport {
2394   \peek_charcode_remove:NTF * {
2395     \gimport_do:
2396   } {
2397     \gimport_do:
2398   }
2399 }
2400
2401 \NewDocumentCommand \gimport_do: { O{} m } {
2402   \msg_set:nnn{stex}{warning/deprecated}{
2403     \\\
2404     \c_backslash_str gimport~is~deprecated! \\\
2405     Please~use~\c_backslash_str importmodule[#1]{#2}~instead!~(in~file~
2406     \stex_path_to_string:N \g_stex_currentfile_seq)
2407     \\\ \\\
2408   }
2409   \msg_warning:nn{stex}{warning/deprecated}
2410   \importmodule[#1]{#2}
2411 }
2412
2413 \cs_new_protected:Npn \guse {
2414   \peek_charcode_remove:NTF * {
2415     \guse_do:
2416   } {
2417     \guse_do:
2418   }
2419 }
2420
2421 \NewDocumentCommand \guse_do: { O{} m } {
2422   \msg_set:nnn{stex}{warning/deprecated}{
2423     \\\
2424     \c_backslash_str guse~is~deprecated! \\\
2425     Please~use~\c_backslash_str usemodule[#1]{#2}~instead!~(in~file~
2426     \stex_path_to_string:N \g_stex_currentfile_seq)
2427     \\\ \\\
2428   }
2429   \msg_warning:nn{stex}{warning/deprecated}
2430   \usemodule[#1]{#2}
2431 }
2432
2433 \cs_new_protected:Npn \symi {
2434   \peek_charcode_remove:NTF * {
2435     \symi_do:
2436   } {
2437     \symi_do:
2438   }
2439 }
2440

```

```

2441 \NewDocumentCommand \symi_do: { 0{} m } {
2442   \msg_set:nnn{stex}{warning/deprecated}{
2443     \\\
2444     \c_backslash_str symi~is~deprecated! \\\
2445     Please~use~\c_backslash_str symdecl{#1}{#2}~instead!~(in~file~
2446     \stex_path_to_string:N \g_stex_currentfile_seq)
2447     \\\ \\\
2448   }
2449   \msg_warning:nn{stex}{warning/deprecated}
2450   \symdecl*{#1}{#2}
2451 }
2452
2453 \cs_new_protected:Npn \symii {
2454   \peek_charcode_remove:NTF * {
2455     \symii_do:
2456   } {
2457     \symii_do:
2458   }
2459 }
2460
2461 \NewDocumentCommand \symii_do: { 0{} m m } {
2462   \msg_set:nnn{stex}{warning/deprecated}{
2463     \\\
2464     \c_backslash_str symii~is~deprecated! \\\
2465     Please~use~\c_backslash_str symdecl{#1}{#2-#3}~instead!~(in~file~
2466     \stex_path_to_string:N \g_stex_currentfile_seq)
2467     \\\ \\\
2468   }
2469   \msg_warning:nn{stex}{warning/deprecated}
2470   \symdecl*{#1}{#2-#3}
2471 }
2472
2473 \cs_new_protected:Npn \symiii {
2474   \peek_charcode_remove:NTF * {
2475     \symiii_do:
2476   } {
2477     \symiii_do:
2478   }
2479 }
2480
2481 \NewDocumentCommand \symiii_do: { 0{} m m m } {
2482   \msg_set:nnn{stex}{warning/deprecated}{
2483     \\\
2484     \c_backslash_str symiii~is~deprecated! \\\
2485     Please~use~\c_backslash_str symdecl{#1}{#2-#3-#4}~instead!~(in~file~
2486     \stex_path_to_string:N \g_stex_currentfile_seq)
2487     \\\ \\\
2488   }
2489   \msg_warning:nn{stex}{warning/deprecated}
2490   \symdecl*{#1}{#2-#3-#4}
2491 }
2492
2493 \keys_define:nn { stex / deprec / defi } {
2494   name .tl_set_x:N = \l_tmpa_str

```

```

2495 }
2496
2497 \cs_new_protected:Npn \defi {
2498   \peek_charcode_remove:NTF * {
2499     \defi_do:
2500   } {
2501     \defi_do:
2502   }
2503 }
2504
2505 \NewDocumentCommand \defi_do: { 0{ } m } {
2506   \str_clear:N \l_tmpa_str
2507   \keys_set:nn { stex / deprec / defi } { #1 }
2508
2509   \str_if_empty:NTF \l_tmpa_str {
2510     \msg_set:nnn{stex}{warning/deprecated}{
2511       \\\
2512       \c_backslash_str defi-is-deprecated! \\\
2513       Please~use~\c_backslash_str STEXsymbol{#2}![#2]~instead!~(in~file~
2514       \stex_path_to_string:N \g_stex_currentfile_seq)
2515       \\\ \\\
2516     }
2517     \msg_warning:nn{stex}{warning/deprecated}
2518     \STEXsymbol { #2 }![ \comp{#2} ]
2519   } {
2520     \msg_set:nnn{stex}{warning/deprecated}{
2521       \\\
2522       \c_backslash_str defi-is-deprecated! \\\
2523       Please~use~\c_backslash_str STEXsymbol { \l_tmpa_str }[ #2 ]~instead!~(in~file~
2524       \stex_path_to_string:N \g_stex_currentfile_seq)
2525       \\\ \\\
2526     }
2527     \msg_warning:nn{stex}{warning/deprecated}
2528     \exp_args:No \STEXsymbol { \l_tmpa_str }[ \comp{#2} ]
2529   }
2530 }
2531
2532 \cs_new_protected:Npn \defii {
2533   \peek_charcode_remove:NTF * {
2534     \defii_do:
2535   } {
2536     \defii_do:
2537   }
2538 }
2539
2540 \NewDocumentCommand \defii_do: { 0{ } m m } {
2541   \str_set:Nn \l_tmpa_str { #1 }
2542   \str_if_empty:NTF \l_tmpa_str {
2543     \msg_set:nnn{stex}{warning/deprecated}{
2544       \\\
2545       \c_backslash_str defii-is-deprecated! \\\
2546       Please~use~\c_backslash_str STEXsymbol{#2~#3}![#2~#3]~instead!~(in~file~
2547       \stex_path_to_string:N \g_stex_currentfile_seq)
2548       \\\ \\\

```



```

2549 }
2550 \msg_warning:nn{stex}{warning/deprecated}
2551 \STEXsymbol { #2-#3 }![ \comp{#2~#3} ]
2552 } {
2553 \msg_set:nnn{stex}{warning/deprecated}{
2554   \
2555   \c_backslash_str defii~is~deprecated! \
2556   Please~use~\c_backslash_str STEXsymbol { #1 ? #2-#3 }[ #2~#3 ]~instead!~(in~file~
2557   \stex_path_to_string:N \g_stex_currentfile_seq)
2558   \
2559 }
2560 \msg_warning:nn{stex}{warning/deprecated}
2561 \STEXsymbol { #1 ? #2-#3 }[ \comp{#2~#3} ]
2562 }
2563 }
2564
2565
2566 \cs_new_protected:Npn \defiis {
2567   \peek_charcode_remove:NTF * {
2568     \defiis_do:
2569   } {
2570     \defiis_do:
2571   }
2572 }
2573
2574 \NewDocumentCommand \defiis_do: { O{} m m } {
2575   \str_set:Nn \l_tmpa_str { #1 }
2576   \str_if_empty:NTF \l_tmpa_str {
2577     \msg_set:nnn{stex}{warning/deprecated}{
2578       \
2579       \c_backslash_str defiis~is~deprecated! \
2580       Please~use~\c_backslash_str STEXsymbol{#2-#3}![#2~#3s]~instead!~(in~file~
2581       \stex_path_to_string:N \g_stex_currentfile_seq)
2582       \
2583     }
2584     \msg_warning:nn{stex}{warning/deprecated}
2585     \STEXsymbol { #2-#3 }![ \comp{#2~#3s} ]
2586   } {
2587     \msg_set:nnn{stex}{warning/deprecated}{
2588       \
2589       \c_backslash_str defii~is~deprecated! \
2590       Please~use~\c_backslash_str STEXsymbol { #1 ? #2-#3 }[ #2~#3s ]~instead!~(in~file~
2591       \stex_path_to_string:N \g_stex_currentfile_seq)
2592       \
2593     }
2594     \msg_warning:nn{stex}{warning/deprecated}
2595     \STEXsymbol { #1 ? #2-#3 }[ \comp{#2~#3s} ]
2596   }
2597 }
2598
2599 %\RequirePackage[hyperref]{ntheorem}
2600 %\theoremstyle{plain}
2601 %\RequirePackage{amsthm}
2602

```

```

2603 \NewDocumentEnvironment {definition} { 0{} } {
2604   \stex_smsmode_set_codes:
2605   \msg_set:nnn{stex}{warning/deprecated}{
2606     \\\
2607     definition~environment~is~deprecated!~(in~file~
2608     \stex_path_to_string:N \g_stex_currentfile_seq)
2609     \\\ \\\
2610   }
2611   \msg_warning:nn{stex}{warning/deprecated}
2612 }{}
2613
2614 \NewDocumentCommand \trefi { 0{} m } {
2615   \str_set:Nn \l_tmpa_str { #1 }
2616   \str_if_empty:NTF \l_tmpa_str {
2617     \msg_set:nnn{stex}{warning/deprecated}{
2618       \\\
2619       \c_backslash_str trefi~is~deprecated! \\\
2620       Please~use~\c_backslash_str STEXsymbol{#2}![#2]~instead!~(in~file~
2621       \stex_path_to_string:N \g_stex_currentfile_seq)
2622       \\\ \\\
2623     }
2624     \msg_warning:nn{stex}{warning/deprecated}
2625     \STEXsymbol { #2 }![ \comp{#2} ]
2626   } {
2627     \msg_set:nnn{stex}{warning/deprecated}{
2628       \\\
2629       \c_backslash_str trefi~is~deprecated! \\\
2630       Please~use~\c_backslash_str STEXsymbol { #1 ? #2 }[ #2 ]~instead!~(in~file~
2631       \stex_path_to_string:N \g_stex_currentfile_seq)
2632       \\\ \\\
2633     }
2634     \msg_warning:nn{stex}{warning/deprecated}
2635     \STEXsymbol { #1 ? #2 }[ \comp{#2} ]
2636   }
2637 }
2638
2639 \NewDocumentCommand \symvariant { 0{} m 0{0} m m } {
2640   \msg_set:nnn{stex}{warning/deprecated}{
2641     \\\
2642     \c_backslash_str symvariant~is~deprecated! \\\
2643     Please~use~\c_backslash_str notation[#4]{ #2 }~instead!~(in~file~
2644     \stex_path_to_string:N \g_stex_currentfile_seq)
2645     \\\ \\\
2646   }
2647   \msg_warning:nn{stex}{warning/deprecated}
2648
2649   \notation[variant=#4]{#2}{#5}
2650 }
2651
2652 \NewDocumentCommand \mixfixi { 0{} m m m } {
2653   \msg_set:nnn{stex}{warning/deprecated}{
2654     \c_backslash_str mixfixi~is~fatally~deprecated!\\
2655     Symbol:~\l_stex_term_highlight_uri_str\\
2656     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq

```

```

2657 }
2658 \msg_error:nn{stex}{warning/deprecated}
2659 }
2660
2661
2662 \NewDocumentCommand \infix {} {
2663   \msg_set:nnn{stex}{warning/deprecated}{
2664     \c_backslash_str infix~is~fatally~deprecated!\\
2665     Symbol:~\l__stex_term_highlight_uri_str\\
2666     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
2667   }
2668   \msg_error:nn{stex}{warning/deprecated}
2669 }
2670
2671 \let\iprec\infprec
2672
2673 \NewDocumentCommand \inlineex { m } {
2674   \msg_set:nnn{stex}{warning/deprecated}{
2675     \c_backslash_str inlineex~is~deprecated!\\
2676     No~replacement~exists~yet.\\
2677     Current~file:~\stex_path_to_string:N \g_stex_currentfile_seq
2678   }
2679   \msg_warning:nn{stex}{warning/deprecated}
2680   #1
2681 }
2682
2683
2684 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { mhmodnl } }
2685 \seq_gput_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n { modsig } }
2686 \tl_gput_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\gimport\syml\symii\symiii\symiv\
2687
2688 </compat>

```