

The tokglobalstack Package

Version 1.0

Alceu Frigeri*

February 2026

Abstract

This package offers two stack’s implementations. Those stacks can be used to, for instance, preserve some tokens beyond a variable number of nested groups, or to implement recursive commands without relying on local groups.

Contents

1	Introduction	1
2	Stack Variable	1
3	Custom Stack Commands	2
4	Examples of Use	2
4.1	Using Custom Stack Commands	2
4.2	Using a Stack Variable	3

1 Introduction

When it’s necessary to preserve the value of some tokens beyond a local group, in simple cases, it’s enough to use `\group_insert_after:N` or `\aftergroup`.

But, sometimes you don’t have (or can’t use) this information, see [1], for instance. For those cases, this package deploys a strategy (presented and compared in [2]) based on stacks.

These stacks can also be used to write recursive commands (preserving variables) without having to rely on local group scoping.

2 Stack Variable

<code>\globalstack_new:N</code>	<code>\globalstack_new:N {⟨stack-var⟩}</code>
<code>\globalstack_gpush:Nn</code>	<code>\globalstack_gpush:Nn {⟨stack-var⟩}{⟨tokens⟩}</code>
<code>\globalstack_gput_right:Nn</code>	<code>\globalstack_gput_right:Nn {⟨stack-var⟩}{⟨tokens⟩}</code>
<code>\globalstack_gput_left:Nn</code>	<code>\globalstack_gput_left:Nn {⟨stack-var⟩}{⟨tokens⟩}</code>
<code>\globalstack_gpop:N</code>	<code>\globalstack_gpop:N {⟨stack-var⟩}</code>

`\globalstack_new:N` will globally create a stack variable named `⟨stack-var⟩` (a specialized token list variable). Once created it is possible to push tokens into it (`\globalstack_gpush:Nn`), amend tokens to the top (`\globalstack_gput_right:Nn` and `\globalstack_gput_left:Nn`) and pop those tokens (`\globalstack_gpop:N`) into the input stream. All assignments being global.

Note: An error will be raised if `⟨stack-var⟩` is already used.

*<https://github.com/alceu-frigeri/tokglobalstack>

3 Custom Stack Commands

```
\globalstack_csnew:n \globalstack_csnew:n {(stack-prefix)}
```

This will globally create a set of commands, named after $\langle \text{stack-prefix} \rangle$, to push, put and pop items from a private global stack. All assignments to/from that stack will be global, and the stack itself will be unique to the command's set.

Note: An error will be raised if $\langle \text{stack-prefix} \rangle$ is already used.

```
\<stack-prefix>_gpush:n \<stack-prefix>_gpush:n {(tokens)}
\<stack-prefix>_gput_right:n \<stack-prefix>_gput_right:n {(tokens)}
\<stack-prefix>_gput_left:n \<stack-prefix>_gput_left:n {(tokens)}
\<stack-prefix>_gpop: \<stack-prefix>_gput_gpop:
```

The $\backslash \langle \text{stack-prefix} \rangle _ \text{gpush:n}$ will push $\langle \text{tokens} \rangle$ (can be any number of tokens) into a global, private, stack. $\backslash \langle \text{stack-prefix} \rangle _ \text{gput_right:n}$ and $\backslash \langle \text{stack-prefix} \rangle _ \text{gput_left:n}$ will amend tokens to it, and $\backslash \langle \text{stack-prefix} \rangle _ \text{gpob:}$, as the name implies, will insert the top of the stack into the input stream. That way it is possible to have a very fine control of what, where and when the items are collected and used.

4 Examples of Use

In the following examples, two stacks will be used, $\langle \text{myStackA} \rangle$ and $\langle \text{myStackB} \rangle$ (one of each kind).

```
\ExplSyntaxOn
% Just a set of booleans for testing
\bool_new:N \l__mytest_tmpa_bool
\bool_new:N \l__mytest_tmpb_bool
\bool_new:N \l__mytest_tmpc_bool
\cs_new:Npn \mytest_show_bools:n #1
{ \underline{#1:}\par
  \bool_if:NTF \l__mytest_tmpa_bool {{\color{red}a-true}}{a-false} ---
  \bool_if:NTF \l__mytest_tmpb_bool {{\color{red}b-true}}{b-false} ---
  \bool_if:NTF \l__mytest_tmpc_bool {{\color{red}c-true}}{c-false} \par
}

\globalstack_csnew:n {myStackA}
\globalstack_new:N \g_myStackB_stack
\ExplSyntaxOff
```

4.1 Using Custom Stack Commands

Using just one position (of the stack) and restoring all tokens in a single point.

<pre>\ExplSyntaxOn \group_begin: {{ \myStackA_gpush:n {\bool_set_true:N \l__mytest_tmpa_bool} {{ \myStackA_gput_right:n {\bool_set_true:N \l__mytest_tmpb_bool} \mytest_show_bools:n {T1} }} \mytest_show_bools:n {T2} }} \myStackA_gpop: \mytest_show_bools:n {T3} \group_end: \ExplSyntaxOff</pre>	<p><u>T1:</u> a false - b false - c false</p> <p><u>T2:</u> a false - b false - c false</p> <p><u>T3:</u> a true - b true - c false</p>
--	--

Using two positions (of the stack) and restoring the tokens in separated points.

<pre> \ExplSyntaxOn \group_begin: {{ \myStackA_gpush:n {\bool_set_true:N \l__mytest_tmpa_bool} }} {{ \myStackA_gpush:n {\bool_set_true:N \l__mytest_tmpb_bool} \mytest_show_bools:n {T1} }} {{ \myStackA_gpop: \mytest_show_bools:n {T2} }} {{ \myStackA_gpop: \mytest_show_bools:n {T3} }} \group_end: \ExplSyntaxOff </pre>	<p><u>T1:</u> a false - b false - c false</p> <p><u>T2:</u> a false - b true - c false</p> <p><u>T3:</u> a true - b false - c false</p>
---	---

4.2 Using a Stack Variable

Using just one position (of the stack) and restoring all tokens in a single point.

<pre> \ExplSyntaxOn \group_begin: {{ \globalstack_gpush:Nn \g_myStackB_stack {\bool_set_true:N \l__mytest_tmpa_bool} }} {{ \globalstack_gput_right:Nn \g_myStackB_stack {\bool_set_true:N \l__mytest_tmpb_bool} \mytest_show_bools:n {T1} }} \mytest_show_bools:n {T2} {{ \globalstack_gpop:N \g_myStackB_stack \mytest_show_bools:n {T3} }} \group_end: \ExplSyntaxOff </pre>	<p><u>T1:</u> a false - b false - c false</p> <p><u>T2:</u> a false - b false - c false</p> <p><u>T3:</u> a true - b true - c false</p>
--	---

Using two positions (of the stack) and restoring the tokens in separated points.

<pre> \ExplSyntaxOn \group_begin: {{ \globalstack_gpush:Nn \g_myStackB_stack {\bool_set_true:N \l__mytest_tmpa_bool} }} {{ \globalstack_gpush:Nn \g_myStackB_stack {\bool_set_true:N \l__mytest_tmpb_bool} \mytest_show_bools:n {T1} }} {{ \globalstack_gpop:N \g_myStackB_stack \mytest_show_bools:n {T2} }} {{ \globalstack_gpop:N \g_myStackB_stack \mytest_show_bools:n {T3} }} \group_end: \ExplSyntaxOff </pre>	<p><u>T1:</u> a false - b false - c false</p> <p><u>T2:</u> a false - b true - c false</p> <p><u>T3:</u> a true - b false - c false</p>
---	---

References

- [1] David Carlisle. *Stackexchange about grouping*. 2026. URL: https://tex.stackexchange.com/questions/757755/coffins-scope-groups#comment1889872_757755 (visited on 01/01/2026).
- [2] Alceu Frigeri. *The xstacks package*. 2026. URL: <https://ctan.org/pkg/xstacks> (visited on 02/18/2026).