

# YAML Data Serialization Support

Operation	Keystroke	Function	Note																								
<b>Editing YAML files</b> ○ Help & Customization • Syntax check • Highlight Column • Indentation / Text Folding • indent-tools ○ Smartparens • smart-shift ○ YAML Reference		Emacs supports YAML via the built-in <code>yaml-ts-mode</code> a <a href="#">Tree Sitter</a> mode that PEL supports on <b>Emacs &gt;= 30.1</b> only. You can also use <code>yaml-mode</code> external package with provides a classic major-mode that does not require Tree-Sitter support. <ul style="list-style-type: none"><li>• PEL installs and activates it when the PEL user-option below is set.</li></ul>	See also: <a href="#">M_CWL</a>																								
	 <code>yml-mode</code>	 <code>pel-use-yaml</code>	A classic major mode for YAML files. Set user-option to <code>t</code> to install and activate.																								
		Long YAML files are notoriously difficult to edit properly. The following packages provide useful minor-modes for working with YAML files:																									
	 <code>flycheck</code>	 <code>pel-use-flycheck</code>	Supports checking the syntax of the YAML file.																								
	 <code>indent-tools</code>	 <code>pel-use-indent-tools</code>	Tools to indent, unindent, navigate across indentation levels																								
	 <code>smartparens</code>	 <code>pel-use-smart-parens</code>	Provide commands to manipulate code in matching pairs.																								
	 <code>smart-shift</code>	 <code>pel-use-smart-shift</code>	Provide commands to shift (indent/un-indent) block of lines.																								
	 <code>vline.el</code>	 <code>pel-use-vline</code>	A command that display a vertical line: helps with alignment.																								
		 <code>pel-yaml-activates-minor-modes</code>	Identify the minor modes to activate automatically inside YAML buffers using classic or Tree-Sitter major mode.																								
Last updated on:	2026-01-09	• PEL associates the following file extensions with <code>yaml-mode</code> : <code>.yml</code> , <code>.yaml</code> , <code>.eyaml</code> , <code>.raml</code> .																									
<b>Open this PDF file.</b> See also: <a href="#">Help/Info</a>	<code>&lt;f11&gt; SPC M-y &lt;f1&gt;</code>	( <code>pel-help-pdf</code> &optional OPEN-WEB-PAGE)	Open the <a href="#">YAML</a> local PDF. If the prefix argument (like <code>C-u</code> or <code>M--</code> ) is used, then it opens the remote GitHub hosted raw PDF instead. If the <code>pel-flip-help-pdf-arg</code> user-option is set it's the other way around.																								
<a href="#">Customize PEL YAML control</a>	<code>&lt;f11&gt; SPC M-y &lt;f2&gt;</code>	( <code>pel-customize-pel</code> &optional OTHER-WINDOW)	Customize PEL YAML support. <ul style="list-style-type: none"><li>• If OTHER-WINDOW is non-nil (use <code>C-u</code>), display in other window.</li></ul>																								
<a href="#">Customize Emacs YAML control</a>	<code>&lt;f11&gt; SPC M-y &lt;f3&gt;</code>	( <code>pel-customize-library</code> &optional OTHER-WINDOW)	Customize Emacs YAML support groups: yaml, fly check, indent-tools and smartparens <ul style="list-style-type: none"><li>• If OTHER-WINDOW is non-nil (use <code>C-u</code>), display in other window.</li></ul>																								
<a href="#">Show PEL setup information for the major mode.</a>	<code>&lt;f11&gt; ? /</code>	( <code>pel-mode-setup-info</code> &optional APPEND)	Display yaml-mode setup information inside a *pel-mode-info* buffer with buttons providing quick access to the customization buffer of each variable shown. <ul style="list-style-type: none"><li>• Use any prefix key (such as <code>C-u</code>) too append information in the buffer instead of clearing it.</li></ul>																								
Toggle between classic and <a href="#">Tree Sitter</a> major mode	<code>&lt;f11&gt; C-t C-t</code>	( <code>pel-treesit-toggle-mode</code> )	Toggle the major mode between the classic mode and the Tree-Sitter based mode. <ul style="list-style-type: none"><li>• If the other major mode is not available the command signals a user error.</li></ul>																								
<b>Flycheck</b>		 activated by PEL <code>pel-use-flycheck</code> is a minor mode for on-the-fly syntax checking..  Aside from the following 2 key bindings that PEL provides to toggle the flycheck mode, flycheck key prefix is <code>C-c !</code> as set by its <code>flycheck-keymap-prefix</code> user-option. You can change it for a different key prefix.																									
See also: <a href="#">SyntaxCheck</a>																											
Toggle flycheck mode for current buffer	<code>&lt;f11&gt; ! !</code>	( <code>flycheck-mode</code> &optional ARG)	Toggle flycheck minor-mode for the current buffer.																								
Toggle flycheck mode for all buffers	<code>&lt;f11&gt; ! M-!</code>	( <code>global-flycheck-mode</code> &optional ARG)	Toggle Flycheck mode in all buffers. <ul style="list-style-type: none"><li>• Flycheck mode is enabled in all buffers where 'flycheck-mode-on-safe' would do it.</li></ul>																								
• <b>Flycheck buffer/file</b>																											
Syntax Check current buffer	<code>C-c ! c</code>	( <code>flycheck-buffer</code> )	Start checking syntax in the current buffer. <ul style="list-style-type: none"><li>• Get a syntax checker for the current buffer with 'flycheck-get-checker-for-buffer', and start it.</li></ul>																								
Check syntax of current file	<code>C-c ! c-c</code>	( <code>flycheck-compile</code> CHECKER)	Run CHECKER via 'compile'. <ul style="list-style-type: none"><li>• CHECKER must be a valid syntax checker. Interactively, prompt for a syntax checker to run.</li><li>• Instead of highlighting errors in the buffer, this command pops up a separate buffer with the entire output of the syntax checker tool, just like 'compile'.</li></ul>																								
<b>Highlight current column</b>		The following command provide a vertical line across the entire window at the cursor location. <ul style="list-style-type: none"><li>• Useful when creating tables or checking indentation manually.</li><li>• vline also provides the vline-global-mode to activate the vertical line in all buffers; PEL has no binding for it because it slows Emacs too much.</li></ul>																									
<b>Toggle Vline Mode</b> See also: <a href="#">Highlight</a>	• <code>&lt;f11&gt; h  </code> • <code>&lt;f11&gt; 9</code>	( <code>vline-mode</code> &optional ARG)	Toggle the display of a vertical line spanning the entire window at the cursor column.  Requires: <code>vline.el</code>  PEL activates it when <code>pel-use-vline</code> user option is <code>t</code> .																								
<b>Indented Text Folding</b>		The following command folds (hide or show) all lines that are indented more than the current line. <ul style="list-style-type: none"><li>• You can also use the <code>f</code> key inside the indent-tools Hydra, shown below, to fold indented sections.</li></ul>																									
Toggle hiding lines more indented than current line	<code>&lt;f11&gt; H I</code>	( <code>pel-toggle-hide-indent</code> )	Toggle hiding lines more indented than current line. <ul style="list-style-type: none"><li>• Affects the entire buffer. Not syntax sensitive. Can be used anywhere.</li></ul>																								
See also: <a href="#">Hide/Show</a>		 Do not modify the buffer while lines are hidden, it's allowed but its using selective display and you don't see what you change.																									
<b>Indent-tools</b>  PEL activates it when the <code>pel-use-indent-tools</code>		The <code>indent-tools</code> external package provides a minor-mode with several commands to indent, un-indent, navigate across indented text levels with a key <code>hydra</code>  PEL provide a global key binding to its key <code>hydra</code> and provides the ability to activate the proposed key binding globally and for python mode: <ul style="list-style-type: none"><li>• <code>pel-indent-tools-key-bound</code> : activates the <code>C-c &gt;</code> key binding either globally or for python-mode only.</li></ul>																									
Open the indent-tools hydra	• <code>&lt;f11&gt; &lt;tab&gt; &lt;f7&gt;</code> • <code>&lt;f7&gt; &lt;tab&gt;</code> • <code>C-c &gt;</code>	( <code>indent-tools-hydra/body</code> )	Activate the body in the "indent-tools-hydra" hydra.  With PEL, these key bindings are only available when: globally, when <code>pel-indent-tools-key-bound</code> is set to <code>globally</code> , <ul style="list-style-type: none"><li>• in python-mode only when <code>pel-indent-tools-key-bound</code> is set to <code>python</code>.</li><li>• The actual key is selected by indent-tools <code>indent-tools-keymap-prefix</code> user-option, the default is <code>C-c &gt;</code></li></ul>																								
The indent-tools hydra provide keys you can use to navigate across the indented YAML elements.		The heads for the associated hydra are: <code>&gt;:</code> 'indent-tools-indent', <code>&lt;:</code> 'indent-tools-demote', <code>E:</code> 'indent-tools-indent-end-of-defun', <code>c:</code> 'indent-tools-comment', <code>U:</code> 'indent-tools-uncomment', <code>P:</code> 'indent-tools-indent-paragraph', <code>l:</code> 'indent-tools-indent-end-of-level', <code>K:</code> 'indent-tools-kill-tree', <code>C:</code> 'indent-tools-copy-hydra/body', <code>s:</code> 'indent-tools-select', <code>e:</code> 'indent-tools-goto-end-of-tree', <code>u:</code> 'indent-tools-goto-parent', <code>d:</code> 'indent-tools-goto-child', <code>S:</code> 'indent-tools-select-end-of-tree', <code>n:</code> 'indent-tools-goto-next-sibling', <code>p:</code> 'indent-tools-goto-previous-sibling', <code>i:</code> 'helm-imenu', <code>j:</code> 'forward-line', <code>k:</code> 'previous-line', <code>SPC:</code> 'indent-tools-indent-space', <code>_:</code> 'undo-tree-undo', <code>L:</code> 'recenter-top-bottom', <code>f:</code> 'yaftolding-toggle-element', <code>q:</code> exit	<table border="1"><thead><tr><th>Indent</th><th>Navigation</th><th>Actions</th></tr></thead><tbody><tr><td><code>&gt; indent</code></td><td><code>j v</code></td><td><code>K kill</code></td></tr><tr><td><code>&lt; de-indent</code></td><td><code>k ^</code></td><td><code>i imenu</code></td></tr><tr><td><code>l end of level</code></td><td><code>n next sibling</code></td><td><code>C Copy...</code></td></tr><tr><td><code>E end of fn</code></td><td><code>p previous sibling</code></td><td><code>c comment</code></td></tr><tr><td><code>P paragraph</code></td><td><code>u up parent</code></td><td><code>U uncomment (paragraph)</code></td></tr><tr><td><code>SPC space</code></td><td><code>d down child</code></td><td><code>f fold</code></td></tr><tr><td><code>_ undo</code></td><td><code>e end of tree</code></td><td><code>q quit</code></td></tr></tbody></table>	Indent	Navigation	Actions	<code>&gt; indent</code>	<code>j v</code>	<code>K kill</code>	<code>&lt; de-indent</code>	<code>k ^</code>	<code>i imenu</code>	<code>l end of level</code>	<code>n next sibling</code>	<code>C Copy...</code>	<code>E end of fn</code>	<code>p previous sibling</code>	<code>c comment</code>	<code>P paragraph</code>	<code>u up parent</code>	<code>U uncomment (paragraph)</code>	<code>SPC space</code>	<code>d down child</code>	<code>f fold</code>	<code>_ undo</code>	<code>e end of tree</code>	<code>q quit</code>
Indent	Navigation	Actions																									
<code>&gt; indent</code>	<code>j v</code>	<code>K kill</code>																									
<code>&lt; de-indent</code>	<code>k ^</code>	<code>i imenu</code>																									
<code>l end of level</code>	<code>n next sibling</code>	<code>C Copy...</code>																									
<code>E end of fn</code>	<code>p previous sibling</code>	<code>c comment</code>																									
<code>P paragraph</code>	<code>u up parent</code>	<code>U uncomment (paragraph)</code>																									
<code>SPC space</code>	<code>d down child</code>	<code>f fold</code>																									
<code>_ undo</code>	<code>e end of tree</code>	<code>q quit</code>																									
See also: <a href="#">Hide/Show</a>			 The <code>f</code> key toggles the element. Press once to hide the sub-tree, press-again to display it back.																								

Operation	Keystroke	Function	Note
<b>Smartparens Mode</b> • <a href="#">Smartparens manual</a>  See also: <a href="#">Smartparens</a>	Simplify insertion of matching pairs with the <a href="#">smartparens</a> minor mode. PEL binds a set of keys, described below, to toggle activation of that mode.  This uses the <a href="#">smartparens</a> external package.  PEL activates it when <code>pel-use-smartparens</code> is set to t.		
<b>Help on smartparens</b>	<code>&lt;f11&gt; ( ?</code>	(sp-cheat-sheet &optional ARG)	Generate a cheat sheet of all the smartparens interactive functions. Shows inside Emacs buffer. <ul style="list-style-type: none"> <li>Without a prefix argument, print only the short documentation and examples.</li> <li>With non-nil prefix argument ARG, show the full documentation for each function.</li> <li>You can follow the links to the function or variable help page.</li> <li>To get back to the full list, use M-x help-go-back.</li> <li>You can use 'beginning-of-defun' and 'end-of-defun' to jump to the previous/next entry.</li> <li>Examples are fortified using the 'font-lock-string-face' for better orientation.</li> </ul>
<b>Describe user system</b>	<code>&lt;f11&gt; ( M-?</code>	(sp-describe-system STARTERKIT)	Describe user's system. Prompt for starter kit: Evil, Spacemacs, Vanilla. <ul style="list-style-type: none"> <li>The output of this function can be used in bug reports.</li> </ul>
<b>Toggle smartparens mode</b>	<code>&lt;f11&gt; ( (</code>	(smartparens-mode &optional ARG)	Toggle smartparens mode.
<b>Toggle smartparens-strict mode</b>	<code>&lt;f11&gt; ( )</code>	(smartparens-strict-mode &optional ARG)	Toggle the strict smartparens mode. <ul style="list-style-type: none"> <li>When strict mode is active, 'delete-char', 'kill-word' and their backward variants will skip over the pair delimiters in order to keep the structure always valid (the same way as 'paredit-mode' does). This is accomplished by remapping them to 'sp-delete-char' and 'sp-kill-word'. There is also function 'sp-kill-symbol' that deletes symbols instead of words, otherwise working exactly the same (it is not bound to any key by default).</li> <li>When strict mode is active, this is indicated with "/" after the smartparens indicator in the mode list</li> </ul>
<b>Toggle smartparens mode</b>	<code>&lt;f11&gt; ( M-(</code>	(smartparens-global-mode &optional ARG)	Toggle Smartparens mode in all buffers. <ul style="list-style-type: none"> <li>With prefix ARG, enable Smartparens-Global mode if ARG is positive; otherwise, disable it.</li> <li>Smartparens mode is enabled in all buffers where 'turn-on-smartparens-mode' would do it.</li> </ul>
<b>Toggle smartparens-strict mode</b>	<code>&lt;f11&gt; ( M-)</code>	(smartparens-global-strict-mode &optional ARG)	Toggle Smartparens-Strict mode in all buffers. <ul style="list-style-type: none"> <li>With prefix ARG, enable Smartparens-Global-Strict mode if ARG is positive; otherwise, disable it.</li> <li>Smartparens-Strict mode is enabled in all buffers where 'turn-on-smartparens-strict-mode' would do it.</li> </ul>
<b>Smart-shift</b>  See also: <a href="#">Indentation</a>	The <a href="#">smart-shift</a> external package simplifies shifting a complete line or region of lines right or left but also up or down. <ul style="list-style-type: none"> <li>It is implemented as a minor or global minor mode that must be enabled first. You can identify the <code>smart-shift-mode</code> inside one of the <code>pel-&lt;mode&gt;-activates-minor-modes</code> user-options to activate it automatically. You can also use the commands manually or through the key bindings provided by PEL to activate the <code>smart-shift-mode</code> in the current buffer or globally for all buffers.</li> <li>PEL controls it through customization user-options:  The <a href="#">smart-shift</a> external package  PEL activates it when the <code>pel-use-smart-shift</code> user-option is turned on (set to t).   PEL also provides the <code>pel-smart-shift-keybinding</code> user-option that allows you to select additional alternative key bindings for the smart-shift commands that shift line(s). By default the key bindings are using C-c as a key prefix. With PEL you can also use a control key for the cursor or change the prefix key to use the &lt;f9&gt; key. The 3 possible key bindings are shown below but only one of them will be available at any given time. The one available is the one selected by the user-option value.</li> </ul>		
<b>Toggle smart-shift mode in current buffer</b>	<code>&lt;f11&gt; &lt;tab&gt; s</code>	(smart-shift-mode &optional ARG)	Activate/de-activate the smart-shift mode in the current buffer. <ul style="list-style-type: none"> <li>Activate the line-shift key bindings listed below, in the current buffer.           <ul style="list-style-type: none"> <li>With PEL, the actual key binding selected for the line shift commands depend on the value of the <code>pel-smart-shift-keybinding</code> user-option.</li> </ul> </li> </ul>
<b>Toggle smart-shift mode globally</b>	<code>&lt;f11&gt; &lt;tab&gt; S</code>	(global-smart-shift-mode &optional ARG)	<ul style="list-style-type: none"> <li>Toggle Smart-Shift mode in all buffers.</li> <li>With prefix ARG, enable Global Smart-Shift mode if ARG is positive; otherwise, disable it.</li> <li>Smart-Shift mode is enabled in all buffers where 'smart-shift-mode-on' would do it.</li> </ul>
<b>Shift line or region right</b>	<ul style="list-style-type: none"> <li>C-c &lt;right&gt;</li> <li>C-c C-&lt;right&gt;</li> <li>&lt;f9&gt; &lt;right&gt;</li> </ul>	(smart-shift-right &optional ARG)	Shift the line or region to the ARG times to the right.   With PEL one of the extra key bindings can be enabled via the <code>pel-smart-shift-keybinding</code> user-option. So unlike other cells only one of the last 2 key bindings is available in the smart-shift minor mode.
<b>Shift line or region left</b>	<ul style="list-style-type: none"> <li>C-c &lt;left&gt;</li> <li>C-c C-&lt;left&gt;</li> <li>&lt;f9&gt; &lt;left&gt;</li> </ul>	(smart-shift-left &optional ARG)	Shift the line or region to the ARG times to the left.   With PEL one of the extra key bindings can be enabled via the <code>pel-smart-shift-keybinding</code> user-option. So unlike other cells only one of the last 2 key bindings is available in the smart-shift minor mode.
<b>Shift line or region up</b>	<ul style="list-style-type: none"> <li>C-c &lt;up&gt;</li> <li>C-c C-&lt;up&gt;</li> <li>&lt;f9&gt; &lt;up&gt;</li> </ul>	(smart-shift-up &optional ARG)	Shift the line or region to the ARG times to the upwards.   With PEL one of the extra key bindings can be enabled via the <code>pel-smart-shift-keybinding</code> user-option. So unlike other cells only one of the last 2 key bindings is available in the smart-shift minor mode.
<b>Shift line or region down</b>	<ul style="list-style-type: none"> <li>C-c &lt;down&gt;</li> <li>C-c C-&lt;down&gt;</li> <li>&lt;f9&gt; &lt;down&gt;</li> </ul>	(smart-shift-down &optional ARG)	Shift the line or region to the ARG times to the downwards   With PEL one of the extra key bindings can be enabled via the <code>pel-smart-shift-keybinding</code> user-option. So unlike other cells only one of the last 2 key bindings is available in the smart-shift minor mode.

## YAML & Emacs – References

Description & URL	Notes
<b>YAML</b>	
<a href="#">YAML @ Wikipedia</a>	Overview, syntax, criticisms
<a href="#">YAML official home page</a>	Links to YAML specification, links to various resources and projects. <ul style="list-style-type: none"> <li>• <a href="#">YAML 1.2 Specs</a></li> <li>• <a href="#">YAML 1.1 Specs</a></li> <li>• <a href="#">YAML 1.0 Specs</a></li> </ul>
<a href="#">YAML Resource sites</a>	<ul style="list-style-type: none"> <li>• <a href="#">Learn YAML in Y Minutes</a></li> <li>• Online YAML validator (runs <a href="#">yamlint.py</a>) <span style="color: yellow;">!</span> No link as the site is not using https. Instead install <a href="#">yamlint.py</a> locally and use it on the command line or via Emacs.</li> </ul>
<b>StrictYAML</b>	A stricter, type-safe YAML
<a href="#">StrictYAML @ Github</a>	
<a href="#">StrictYAML @ hitchdev (Python libraries)</a>	
<b>RAML</b>	RESTful API Modeling Language : RAML files have the .raml file extension.
	<ul style="list-style-type: none"> <li>• <a href="#">RAML @ Wikipedia</a></li> <li>• <a href="#">RAML.org</a></li> <li>• <a href="#">RAML Spec @ GitHub</a></li> </ul>
<b>Common Workflow Language</b>	Common Workflow Language (CWL) uses a subset of YAML and provides YAML supporting tools.
See also: <a href="#">M CWL</a>	<ul style="list-style-type: none"> <li>• <a href="#">CWL home page</a> <ul style="list-style-type: none"> <li>• <a href="#">CWL User Guide</a></li> <li>• <a href="#">CWL YAML Guide</a></li> </ul> </li> </ul>
<b>Emacs support for YAML</b>	
<a href="#">yaml-mode (major mode for YAML)</a>	<ul style="list-style-type: none"> <li>• <a href="#">yaml-mode @ GitHub</a></li> <li>• <a href="#">Yaml Mode @ Emacs Wiki</a></li> </ul>
<a href="#">indent-tools</a>	<ul style="list-style-type: none"> <li>• <a href="#">indent-tools @ GitLab</a></li> <li>• <a href="#">indent-tools @ Melpa</a></li> </ul>
<a href="#">smartparens</a>	The smartparens mode can help deal with data that is within matching pair of characters. <ul style="list-style-type: none"> <li>• <a href="#">smartparens @ GitHub</a></li> <li>• <a href="#">smartparens documentation</a></li> </ul>
<b>Emacs/YAML Support Articles</b>	
<a href="#">Blogs about YAML editing on Emacs</a>	<ul style="list-style-type: none"> <li>• <a href="#">The best ways to work with yaml files in Emacs</a>, from Chmouel Boudjnah's blog, 2016-09-07</li> <li>• <a href="#">Editing ansible files in Emacs</a>, from Enis Özgen, 2017-12-29</li> </ul>
<a href="#">General blogs about YAML</a>	<ul style="list-style-type: none"> <li>• <a href="#">10 YAML tips for people who hate YAML</a> <ul style="list-style-type: none"> <li>• BTW, the last tip is: use something else... well... S-expressions are very flexible and powerful.</li> </ul> </li> </ul>
<b>Using YAML on Github</b>	YAML is used to control the actions of several Continuous Integration/Development systems. Github is one of them. <ul style="list-style-type: none"> <li>• <a href="#">Workflow syntax for GitHub Actions</a></li> </ul>