Customizing Sessions Settings

All of the session settings file locations described in this section conform to the XDG Base Directory Specification. They are configurable using environment variables:

Scope	Default	Environment	RStudio Pro/Workbench
User	`~/.config/rstudio`	`XDG_CONFIG_HOME`	`RSTUDIO_CONFIG_HOME`
System	`/etc/rstudio`	`XDG_CONFIG_DIRS`	`RSTUDIO_CONFIG_DIRS`

In accordance with the *Base Directory Specification*, the environment variables specify the location of the `rstudio` folder. For example, to store system-wide preference configuration in `/var/config/rstudio/rstudio-prefs.json`, you would set the `XDG_CONFIG_DIRS` variable to the value `/var/config`.

If specified, the `RSTUDIO` variables take precedence over the `XDG` variables. These variables specify a specific directory (not a base directory). For example, to store system-wide preferences in `/var/config/settings/rstudio-prefs.json`, you would set the `RSTUDIO_CONFIG_DIR` variable to the value `/var/config/settings`.

The examples in this section presume you're setting system-wide settings in `/etc/rstudio`; in each case it's also possible to use a different folder by changing environment variables as described above, or to apply the settings to individual user accounts by changing files in `~/.config/rstudio`.

User preferences

User preferences set in the RStudio Pro's *Global Options* dialog can also be set in the JSON file `rstudio-prefs.json`, located in the settings directory described above.

Schema

The schema for the JSON file can be found at:

```
/usr/lib/rstudio-server/resources/schema/user-prefs-schema.json
```

It documents all of the preferences, shows data types and allowable values, and briefly explains the usage of each. You can see a summary of this information in the Session User Settings appendix.

Example

By default, Posit Workbench only shows the home page (session overview) to users who have multiple sessions running. If you'd like it to be shown to all users regardless of the number of running sessions, set it in the global user preferences file as follows:

Listing 1 `/etc/rstudio/rstudio-prefs.json`

```
{
  "show_user_home_page": "always"
}
```

Snippets

You can install global snippets files for all users in the `/etc/rstudio/snippets` folder. For example, if you'd like to create a snippet `lib` for an R library call:

Listing 2 `/etc/rstudio/snippets/r.snippets`

```
snippet lib
    library(${1:package})
```

You can also define snippets for CSS files in the file `css.snippets`, and so on. You can find documentation on the snippet file format in the Cloud 9 IDE snippet documentation.

Note that RStudio Pro will not merge snippet files, which implies the following:

- If you define your own snippets (for a given file type), they will replace those that ship with RStudio Pro (for that same file type).
- If users define their own snippets (for a given file type), changes to the system snippet file (for that same file type) won't have any effect on those users.

Default document templates

RStudio Pro typically opens new documents with completely blank contents. You can, however, define the contents of the blank document by creating a file named `default.X` in `/etc/rstudio/templates`, where `X` is the file extension you wish to customize. For example, to start all R scripts with a standard comment header users can fill out, you could use the following:

Listing 3 `/etc/rstudio/templates/default.R`

```
# ------------------------------------
# Script:
# Author:
# Purpose:
# Notes:
#
# Copyright(c) Corporation Name
# ------------------------------------
```

There are also some special template files which ship with RStudio Pro; these, too, are customizable. In `/etc/rstudio/templates`, you can customize the following:

File	Description
`document.Rmd`	The default R Markdown document file content (without YAML header)
`notebook.Rmd`	The default R Notebook file content (without YAML header)
`presentation.Rmd`	The default R Markdown presentation file content (without YAML header)
`shiny.Rmd`	The default Shiny R Markdown file content (without YAML header)
`query.sql`	The default SQL query

Color themes

You can define additional custom themes for RStudio Pro by placing `.rstheme` files in the following directory:

`/etc/rstudio/themes`

The `.rstheme` file contains plain-text CSS with some special metadata. You can create one by importing an existing TextMate theme file, or by starting from scratch (using an existing theme file as a template). Run the R command `?rstudioapi::addTheme` for more help.

Fonts

RStudio Pro's code editor and R console use a fixed-width font. By default, only fonts that end users have installed locally can be selected. If you wish to make additional fixed-width fonts available to your users, you can place them here:

```
/etc/rstudio/fonts
```

Fonts placed here will be automatically made available for selection in RStudio Pro's Appearances settings (Tools -> Global Options -> Appearance) for all users. It's helpful to place the fonts preferred by your users here because it allows the font to be used in RStudio Pro regardless of what fonts they have installed locally.

The following font formats are supported:

- Web Open Font Format (.woff, .woff2)
- OpenType (.otf)
- Embedded OpenType (.eot)
- TrueType (.ttf)

Only fixed-width fonts are supported by RStudio Pro. Proportional fonts will still be installed, but if users select a proportional font, they will experience cursor positioning problems.

Naming and directory structure

The name of the file is presumed to be the name of the font. If you wish to give the font a custom name, you can place it in a directory with your name of choice. For example:

```
+ fonts/
|
+-- Coding-Font.ttf
|
+-- Coding Font Two/
    |
    +-- CodingFont2-Regular.woff
```

This directory structure would make two fonts available, *Coding-Font* and *Coding Font Two*.

Some fonts come in many different weights and styles. If you want these weights and styles to be treated as single font, you can place them underneath a single folder. This is useful when a theme uses bold or italic variants of a font to decorate code (e.g., to set comments in italics).

To do this, create subfolders with the font's weight or style as the folder's name. For example, this creates a single font, "Coding Font 3", which has two weights (400 and 700 for regular and bold, respectively) and an italic style for each weight.

```
+ fonts/
|
+-- Coding Font Three/
    |
    +-- 400/
    |   |
    |   +-- CodingFont3-Regular.woff
    |   |
    |   +-- italic/
    |       |
    |       +-- CodingFont3-Italic.woff
    |
    +-- 700/
        |
        +-- CodingFont3-Bold.woff
        |
        +-- italic/
            |
            +-- CodingFont3-BoldItalic.woff
```

Autodetection

In addition to displaying a list of fonts installed on the system, RStudio Pro attempts to automatically detect available fixed-width fonts that are installed on a user's browser. For security reasons, it is not possible for RStudio Pro to enumerate all the fonts on the user's browser, so a known list of popular programming fixed-width fonts are checked for compatibility. This list is stored in the option `browser_fixed_width_fonts`.

If your users have a font they prefer but it can't be installed on the system, you can cause RStudio Pro to start scanning for it locally by including it in the set of `browser_fixed_width_fonts` in the global RStudio Pro preferences file, `/etc/rstudio/rstudio-prefs.json`. See User preferences for more information on setting global options.

Keybindings

RStudio Pro keybindings can be globally defined using the following two files:

```
/etc/rstudio/keybindings/editor_commands.json
/etc/rstudio/keybindings/rstudio_commands.json
```

It isn't necessary to hand-author these files; RStudio Pro can generate them for you:

1. Remove the `~/.config/rstudio/keybindings/` folder
2. Start a new RStudio Pro Session and customize the keyboard shortcuts as desired
3. Copy the new `.json` files from `~/.config/rstudio/keybindings` to `/etc/rstudio/keybindings` to make them active for all users on the server

Spelling

You can define additional spelling dictionaries for RStudio Pro by placing dictionary files in the following folders:

Languages

Define additional system languages by placing Hunspell `.aff` files in:

```
/etc/rstudio/dictionaries/languages-system
```

Dictionaries

Define additional custom dictionaries by placing Hunspell`.dic` files in:

```
/etc/rstudio/dictionaries/custom
```