Package 'tinyshinyserver'

January 6, 2026

Type Package

Title Tiny 'shiny' Server - Lightweight Multi-App 'shiny' Proxy

Version 0.1.0

Description A lightweight, 'WebSocket'-enabled proxy server for hosting multiple 'shiny' applications with automatic health monitoring, session management, and resource cleanup. Provides a simple entry point to run the server using a JSON configuration file.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.0)

Imports methods, shiny, callr, jsonlite, later, httr, digest, httpuv, websocket, future, logger, openssl, rmarkdown, quarto

Suggests testthat (>= 3.0.0), roxygen2, devtools, DT, dplyr, flexdashboard, plotly

Config/testthat/edition 3

URL https://github.com/lab1702/tinyshinyserver

BugReports https://github.com/lab1702/tinyshinyserver/issues

RoxygenNote 7.3.3

NeedsCompilation no

Author Lars Bernhardsson [aut, cre]

Maintainer Lars Bernhardsson <cran.sherry228@passinbox.com>

Repository CRAN

Date/Publication 2026-01-06 11:40:02 UTC

Contents

`tinyshinyserver-package` *tinyshinyserver: Tiny Shiny Server - Lightweight Multi-App Shiny Proxy*

Description

A lightweight, WebSocket-enabled proxy server for hosting multiple Shiny applications with automatic health monitoring, session management, and resource cleanup.

Main functions

`start_tss` Start the Tiny Shiny Server with a configuration file

Key features

- Host multiple Shiny applications behind a single proxy
- Resident (always-running) and on-demand application modes
- WebSocket support with session affinity
- Real-time management interface and monitoring
- Automatic health checks and application restart
- Cross-platform support (Windows, Linux, macOS)
- Support for R Markdown and Quarto dashboards

Getting started

1. Install the package: `devtools::install(".")` or similar 2. Load the package: `library(tinyshinyserver)` 3. Copy examples: `file.copy(system.file("examples", package = "tinyshinyserver"), ".", recursive = TRUE)` 4. Start server: `start_tss(config = "examples/config.json")` 5. Access via browser: `http://localhost:3838`

Package resources

The package includes:

- Example Shiny applications in `inst/examples/`
- HTML templates and CSS in `inst/templates/`
- Sample configuration file `inst/examples/config.json`

Use `system.file("examples", package = "tinyshinyserver")` to locate the example files after installation.

Author(s)

Maintainer: Lars Bernhardsson <cran.sherry228@passinbox.com>

See Also

Useful links:

- https://github.com/lab1702/tinyshinyserver
- Report bugs at https://github.com/lab1702/tinyshinyserver/issues

---

`config-format` *Configuration File Format*

Description

Details about the JSON configuration file format used by `start_tss`.

Configuration Structure

The configuration file should be a valid JSON file with the following structure:

```
{
  "apps": [
    {
      "name": "app-name",
      "path": "./path/to/app",
      "resident": true|false
    }
  ],
  "starting_port": 3001,
  "proxy_port": 3838,
  "proxy_host": "127.0.0.1",
  "management_port": 3839,
  "restart_delay": 5,
  "health_check_interval": 10,
  "log_dir": "./logs"
}
```

Required Fields

`apps` Array of Shiny applications to host. Each app must have `name` and `path`.

`starting_port` Starting port number for automatic app port assignment.

`log_dir` Directory where server and application logs will be written.

Optional Fields

`proxy_port` Port for the main proxy server (default: 3838).

`proxy_host` Host interface to bind to (default: "127.0.0.1").

`management_port` Port for the management interface (default: 3839).

`restart_delay` Seconds to wait before restarting failed apps (default: 5).

`health_check_interval` Seconds between health checks (default: 10).

Application Configuration

Each application in the `apps` array can have:

`name` Unique identifier used in URLs and logs. Required.

`path` File system path to the app directory. Required.

`resident` Boolean. If `true`, app runs continuously. If `false` (default), app starts on-demand.

Host Configuration

The `proxy_host` field controls which network interface the server binds to:

- `"127.0.0.1"` or `"localhost"`: Localhost only (most secure)
- `"0.0.0.0"`: All network interfaces (allows external access)
- `"::1"`: IPv6 localhost
- `"::"`: All IPv6 interfaces

Port Assignment

Apps are automatically assigned ports starting from `starting_port`, skipping any reserved ports (`proxy_port` and `management_port`). For example, with `starting_port: 3001`, apps might get ports 3001, 3002, 3003, etc., but will skip 3838 and 3839 if those are the proxy and management ports.

See Also

`start_tss` for starting the server with a configuration file.

Use `system.file("examples", "config.json", package = "tinyshinyserver")` to see a complete example configuration file.

Examples

```
if (interactive()) {
  # Example configuration file:
  config_content <- '
{
  "apps": [
    {
      "name": "dashboard",
      "path": "./apps/dashboard",
      "resident": true
    },
    {
      "name": "reports",
      "path": "./apps/reports",
      "resident": false
    }
  ],
  "starting_port": 3001,
  "proxy_port": 3838,
```

```
  "management_port": 3839,
  "log_dir": "./logs"
}'

  # Write to file and use
  writeLines(config_content, "my-config.json")
  start_tss(config = "my-config.json")
}
```

`example-config` *Example Configuration File*

Description

This provides the path to the example `config.json` file that demonstrates the proper configuration format for `start_tss`. The file includes sample applications and typical server settings.

Format

A JSON file with the standard configuration structure. See `config-format` for details about the configuration format.

Details

Path to the example configuration file included with the package.

The example configuration includes:

- Four sample Shiny applications (sales, inventory, reports, dashboard)
- Mix of resident and on-demand application modes
- Standard port configuration (proxy: 3838, management: 3839)
- Automatic port assignment starting from 3001
- Logging configuration

See Also

- `start_tss` for starting the server
- `config-format` for configuration file format details

Examples

```
if (interactive()) {
  # Get the example configuration file path
  config_file <- system.file("examples", "config.json", package = "tinyshinyserver")

  # View the configuration
  config_content <- readLines(config_file)
```

```
  cat(config_content, sep = "\\n")

  # Copy examples to current directory and start server
  examples_path <- system.file("examples", package = "tinyshinyserver")
  file.copy(examples_path, ".", recursive = TRUE)
  start_tss(config = "examples/config.json")
}
```

`start_tss` *Start Tiny Shiny Server*

Description

Launch the Tiny Shiny Server using a configuration JSON file. This starts a multi-application Shiny server with automatic health monitoring, session management, and WebSocket support.

Usage

```
start_tss(config = "config.json")
```

Arguments

`config`	Character path to a configuration JSON file. Defaults to "config.json" in the current working directory. The configuration file should specify apps, ports, and other server settings.

Details

The server provides:

- A proxy server on the configured port (default 3838)
- A management interface on the configured port (default 3839)
- Automatic port assignment for individual Shiny applications
- Health monitoring and automatic restart for failed apps
- Support for both resident (always-running) and on-demand apps

The configuration file should contain:

- `apps`: Array of Shiny applications with name, path, and resident settings
- `starting_port`: Starting port for auto-assignment to apps
- `proxy_port`: Port for the main proxy server (default 3838)
- `management_port`: Port for the management interface (default 3839)
- `log_dir`: Directory for log files

Access points after starting:

- Main landing page: `http://localhost:3838`
- Management interface: `http://localhost:3839`
- Individual apps: `http://localhost:3838/proxy/{app_name}/`

Value

Invisibly returns the TinyShinyServer instance after starting. The server runs until interrupted (Ctrl-C) or shut down via the management interface.

Examples

```
if (interactive()) {
  library(tinyshinyserver)
  examples_path <- system.file("examples", package = "tinyshinyserver")
  temp_path <- tempdir()
  file.copy(examples_path, temp_path, recursive = TRUE)
  setwd(temp_path)
  start_tss(config = "examples/config.json")
}
```

Index