

Reltool

Copyright © 2009-2017 Ericsson AB, All Rights Reserved
Reltool 0.7
December 1, 2017

Copyright © 2009-2017 Ericsson AB, All Rights Reserved
Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. The Initial Developer of the Original Code is Ericsson AB. Ericsson AB, All Rights Reserved.

December 1, 2017

1 Reltool Users Guide

`Reltool` is a release management tool. It analyses a given Erlang/OTP installation and determines various dependencies between applications. The `graphical` frontend depicts the dependencies and enables interactive customization of a target system. The backend provides a `batch` interface for generation of customized target systems.

1.1 Introduction

`Reltool` is a release management tool. It analyses a given Erlang/OTP installation and determines various dependencies between applications. The `graphical` frontend depicts the dependencies and enables interactive customization of a target system. The backend provides a `batch` interface for generation of customized target systems.

1.1.1 Scope and Purpose

This manual describes the Reltool application, as a component of the Erlang/Open Telecom Platform development environment. It is assumed that the reader is familiar with the Erlang Development Environment, which is described in a separate User's Guide.

1.1.2 Prerequisites

The following prerequisites is required for understanding the material in the Reltool User's Guide:

- familiarity with Erlang/OTP system principles and Erlang/OTP design principles

The application requires Erlang/OTP release R13B02 or later.

1.1.3 About This Manual

In addition to this introductory chapter, the Reltool User's Guide contains the following chapters:

- Chapter 2: "Usage" describes the architecture and typical usage of the application.
- Chapter 3: "Examples" gives some usage examples

1.1.4 Where to Find More Information

Refer to the following documentation for more information about Reltool and about the Erlang/OTP development system:

- the Reference Manual of `Reltool`
- the Erlang/OTP `System Principles`
- the Erlang/OTP `Design Principles`
- Programming Erlang: Software for a Concurrent World (2007), Pragmatic Bookshelf, ISBN13: 9781934356005.

1.2 Usage

1.2.1 Overview

This document focuses on the graphical parts of the tool. The concepts are explained in the reference manual for the module `reltool`.

1.2.2 System window

The system window is started with the function `reltool:start/1`. At startup the tool will process all `beam` files and `app` files in order to find out dependencies between applications and their modules. Once all this information has been derived, it will be possible to explore the tool.

The system window consists of four main pages (tabs):

- Libraries
- System settings
- Applications
- Releases

Click on a name tag to display its page.

Libraries

On the library page it is possible to control which sources the tool will use. The page is organized as a tree which can be expanded and collapsed by clicking on the little symbol in the beginning of the expandable/collapsible lines.

The `Root directory` can be edited by selecting the line where the path of the root directory is displayed and clicking the right mouse button. Choose edit in the menu that pops up.

Library directories can be added, edited or deleted. This is done by selecting the line where the path to a library directory is displayed and clicking the right mouse button. Choose add, edit or delete in the menu that pops up. New library directories can also be added by selecting the line `Library directories` and clicking the right mouse button. Choose add in the menu that pops up.

Escript files can be added, edited or deleted. This is done by selecting the line where the path to an escript file is displayed and clicking the right mouse button. Choose add, edit or delete in the menu that pops up. New escripts can also be added by selecting the line `Escript files` and clicking the right mouse button. Choose add in the menu that pops up.

When libraries and escripts are expanded, the names of their contained applications will be displayed. Double click on an application name to launch an application window.

System settings

On the system settings page it is possible to control some global settings that are used as defaults for all applications. Set the `Application inclusion policy` to `include` to include all applications that are not explicitly excluded. See `incl_cond` (application inclusion) and `mod_cond` (module inclusion) in the reference manual for the module `reltool` for more info.

The system settings page is rather incomplete.

Applications

There are four categories of applications on the applications page. `Included` contains applications that are explicitly included. `Excluded` contains applications that are explicitly excluded. `Derived` contains applications that either are used directly by explicitly included applications or by other derived applications. `Available` contains the remaining applications.

Select one or more applications and click on a button directly below the application column to change application category. For example, select an available application and click on its tick button to move the application to the included category. Clicking on the tick symbol for included applications will move the application back to the available category. The tick is undone.

The symbols in front of the application names are intended to describe the status of the application. There are error and warning symbols to signalize that there is something which needs attention. The tick symbol means that the application is included or derived and no problem has been detected. The cross symbol means that the application is excluded or

available and no problem has been detected. Applications with error symbols are listed first in each category and are followed by the warnings and the normal ones (ticks and crosses) at the end.

Double click on an application to launch its application window.

Releases

The releases page is incomplete and very experimental.

File menu

- `Display application dependency graph` - Launches an application force graph window. All included and derived applications and their dependencies will be shown in a graph.
- `Display module dependency graph` - Launch a module force graph window. All included and derived modules and their dependencies will be shown in a graph.
- `Reset configuration to default`
- `Undo configuration (toggle)`
- `Load configuration` - Loads a new configuration from file.
- `Save configuration` - Saves the current configuration to file. Normally, only the explicit configuration parameters with values that differ from their defaults are saved. But the configuration with or without default values and with or without derived values may also be saved.
- `Generate rel, script and boot files`
- `Generate target system`
- `Close` - Close the system window and all its subwindows.

Dependencies between applications or modules displayed as a graph

The dependency graph windows are launched from the file menu in the system window. The graph depicts all included and derived applications/modules and their dependencies.

It is possible to perform some limited manipulations of the graph. Nodes can be moved, selected, locked or deleted. Move a single node or the entire graph by moving the mouse while the left mouse button is pressed. A node can be locked into a fix position by holding down the shift button when the left mouse button is released. Select several nodes by moving the mouse while the control key and the left mouse button are pressed. Selected nodes can be locked, unlocked or deleted by clicking on a suitable button.

The algorithm that is used to draw a graph with as few crossed links as possible is called force graph. A force graph consists of nodes and directed links between nodes. Each node is associated with a repulsive force that pushes nodes away from each other. This force can be adjusted with the left slider or with the mouse wheel. Each link is associated with an attractive force that pulls the nodes nearer to each other. This force can be adjusted with the right slider. If this force becomes too strong, the graph will be unstable. The third parameter that can be adjusted is the length of the links. It is adjusted with the middle slider.

The `Freeze` button starts/stops the redrawing of the graph. `Reset` moves the graph to the middle of the window and resets all graph settings to default, with the exception of deleted nodes.

1.2.3 Application window

The application window is started by double clicking on an application name. The application window consists of four pages (tabs):

- Application settings
- Modules
- Application dependencies
- Module dependencies

Click on a name tag to display its page.

Application settings

Select version of the application in the `Source selection policy` part of the page. By default the latest version of the application is selected, but it is possible to override this by explicitly selecting another version.

Note that in order for reltool to sort application versions and thereby be able to select the latest, it is required that the version id for the application consits of integers and dots only, for example `1`, `2.0` or `3.17.1`.

By default the `Application inclusion policy` on system level is used for all applications. Set the value to `include` if you want to explicitly include one particular application. Set it to `exclude` if you want to exclude the application despite that it is used by another (explicitly or implicitly) included application. `derived` means that the application automatically will be included if some other (explicitly or implicitly) included application uses it.

By default the `Module inclusion policy` on system level is used for all applications. Set it to `derived` if you only want actually used modules to be included. Set it to `app` if you, besides derived modules, also want the modules listed in the app file to be included. Set it to `ebin` if you, besides derived modules, also want the modules that exist as beam files in the ebin directory to be included. Set it to `all` if you want all modules to be included, that is the union of modules found in the ebin directory and listed in the app file.

The application settings page is rather incomplete.

Modules

There are four categories of modules on the modules page. `Included` contains modules that are explicitly included. `Excluded` contains modules that are explicitly excluded. `Derived` contains modules that either are used directly by explicitly included modules or by other derived modules. `Available` contains the remaining modules.

Select one or more modules and click on a button directly below the module column to change module category. For example, select an available module and click on its tick button to move the module to the included category. Clicking on the tick symbol for included modules will move the module back to the available category. The tick is undone.

The symbols in front of the module names are intended to describe the status of the module. There are error and and warning symbols to signalize that there is something that needs attention. The tick symbol means that the module is included or derived and no problem has been detected. The cross symbol means that the module is excluded or available and no problem has been detected. Modules with error symbols are listed first in each category and are followed by warnings and the normal ones (ticks and crosses) at the end.

Double click on a module to launch its module window.

Application dependencies

There are four categories of applications on the `Application dependencies` page. If the application is used by other applications, these are listed under `Used by`. If the application requires other applications be started before it can be started, these are listed under `Required`. These applications are listed in the `applications` part of the `app` file. If the application includes other applications, these are listed under `Included`. These applications are listed in the `included_applications` part of the `app` file. If the application uses other applications, these are listed under `Uses`.

Double click on an application name to launch an application window.

Module dependencies

There are two categories of modules on the `Module dependencies` page. If the module is used by other modules, these are listed under `Modules using this`. If the module uses other modules, these are listed under `Used modules`.

Double click on an module name to launch a module window.

1.2.4 Module window

The module window is started by double clicking on an module name. The module window consists initially of two pages (tabs):

- Dependencies
- Code

Click on a name tag to display its page.

Dependencies

There are two categories of modules on the `Dependencies` page. If the module is used by other modules, these are listed under `Modules using this`. If the module uses other modules, these are listed under `Used modules`.

Double click on an module name to launch a module window.

Code

On the `Code` page the Erlang source code is displayed. It is possible to search forwards and backwards for text in the module. Enter a regular expression in the `Find` field and press enter. It is also possible to go to a certain line in the module. The `Back` button can be used to go back to the previous position.

Put the marker on a function name and double click to go to the definition of the function. If the function is defined in another module, that module will be loaded and added to the page list.

1.3 Examples

1.3.1 Start and stop windows and servers

The main process in Reltool is the server. It can be used as it is or be used via the GUI frontend process. When the GUI is started, a server process will automatically be started. The GUI process is started with `reltool:start/0`, `reltool:start/1` or `reltool:start_link/1`. The pid of its server can be obtained with `reltool:get_server/1`

```
Erlang R13B02 (erts-5.7.3) [source] [64-bit] [smp:4:4] [rq:4]
                           [async-threads:0] [kernel-poll:false]

Eshell V5.7.3  (abort with ^G)
1> {ok, Win} = reltool:start([]).
{ok,<0.36.01>}
2> {ok, Server} = reltool:get_server([]).
{ok,<0.37.01>}
3> reltool:get_config(Server).
{ok,{sys,[]}}
4> reltool:stop(Win).
ok


5> {ok, Server2} = reltool:start_server([]).
{ok,<0.6535.01>}
6> reltool:get_config(Server2).
{ok,{sys,[]}}
7> reltool:stop(Server2).
ok
```

1.3.2 Inspecting the configuration

```
Erlang R13B02 (erts-5.7.3) [source] [64-bit] [smp:4:4] [rq:4]
                           [async-threads:0] [kernel-poll:false]

Eshell V5.7.3  (abort with ^G)
1> Config = {sys, [{escript,
                    "examples/display_args",
                    [{incl_cond, include}]},
     {app, inets, [{incl_cond, include}]},
     {app, mnesia, [{incl_cond, exclude}]},
     {app, ssl, [{incl_cond, exclude}]},
     {app, runtime_tools, [{incl_cond, exclude}]},
     {app, syntax_tools, [{incl_cond, exclude}]}]}.
{sys,[{escript,"examples/display_args",[{incl_cond,include}]},
      {app,inets,[{incl_cond,include}]},
      {app,mnesia,[{incl_cond,exclude}]},
      {app,ssl,[{incl_cond,exclude}]},
      {app,runtime_tools,[{incl_cond,exclude}]},
      {app,syntax_tools,[{incl_cond,exclude}]}]}



2> {ok, Server} = reltool:start_server([Config]).
{ok,<0.35.0>}
3> reltool:get_config(Server).
{ok,{sys,[{escript,"/clearcase/otp/tools/reltool/examples/display_args",
                   [{incl_cond,include}]}]}}
4> reltool:get_config(Server, false, false).
{ok,{sys,[{escript,"/clearcase/otp/tools/reltool/examples/display_args",
                   [{incl_cond,include}]}]}}



5> reltool:get_config(Server, true, false).
{ok,{sys,[{root_dir,"/ldisk/hakan/otp_test"},
          {lib_dirs,[]},
          {escript,"/clearcase/otp/tools/reltool/examples/display_args",
                   [{incl_cond,include}]},
          {mod_cond,all},
          {incl_cond,derived},
          {boot_rel,"start_clean"},
          {emu_name,"beam"},
          {relocatable,true},
          {profile,development},
          {incl_sys_files,[".*"]},
          {excl_sys_files,[]},
          {incl_app_files,[".*"]},
          {excl_app_files,[]},
          {incl_archive_dirs,[".*"]},
          {excl_archive_dirs,["^include$","^priv$"]},
          {archive_opts,[]},
          {app_type,permanent},
          {app_file,keep},
          {debug_info,keep}]}}



6> reltool:get_config(Server, true, true).
{ok,{sys,[{root_dir,"/ldisk/hakan/otp_test"},
          {lib_dirs,[]},
          {escript,"/clearcase/otp/tools/reltool/examples/display_args",
                   [{incl_cond,include}]},
```

```
          {mod_cond,all},
          {incl_cond,derived},
          {erts,[{vsn,"5.7.3"},
                 {mod,erl_prim_loader,[]},
                 {mod,erlang,[]},
                 {mod,init,[]},
                 {mod,otp_ring0,[]},
                 {mod,prim_file,[]},
                 {mod,prim_inet,[]},
                 {mod,prim_zip,[]},
                 {mod,zlib,[]}]},
          {app,compiler,
               [{vsn,"4.6.3"},
                {mod,beam_asm,[]},
                {mod,beam_block,[]},
                {mod,beam_bool,[]},
                {mod,beam_bsm,[]},
                {mod,beam_clean,[]},
                {mod,beam_dead,[]},
                {mod,beam_dict,[]},
                {mod,beam_disasm,[]},
                {mod,beam_flatten,[]},
                {mod,beam_jump,[]},
                {mod,beam_listing,[]},
                {mod,beam_opcodes,...},
                {mod,...},
                {...}|...]},
          {app,crypto,
               [{vsn,"1.6.1"},
                {mod,crypto,[]},
                {mod,crypto_app,[]},
                {mod,crypto_server,[]},
                {mod,crypto_sup,[]}]},
          {app,hipe,
               [{vsn,"3.7.3"},
                {mod,cerl_cconv,[]},
                {mod,cerl_closurean,[]},
                {mod,cerl_hipeify,[]},
                {mod,cerl_hybrid_transform,[]},
                {mod,cerl_lib,[]},
                {mod,cerl_messagean,[]},
                {mod,cerl_pmatch,[]},
                {mod,cerl_prettypr,[]},
                {mod,cerl_to_icode,[]},
                {mod,cerl_typean,...},
                {mod,...},
                {...}|...]},
          {app,kernel,
               [{vsn,"2.13.3"},
                {mod,application,[]},
                {mod,application_controller,[]},
                {mod,application_master,[]},
                {mod,application_starter,[]},
                {mod,auth,[]},
                {mod,code,[]},
                {mod,code_server,[]},
                {mod,disk_log,[]},
                {mod,disk_log_1,...},
                {mod,...},
                {...}|...]},
          {app,stdlib,
               [{vsn,"1.16.3"},
                {mod,array,[]},
                {mod,base64,[]},
                {mod,beam_lib,[]},
```

```
                {mod,c,[]},
                {mod,calendar,[]},
                {mod,dets,[]},
                {mod,dets_server,[]},
                {mod,dets_sup,...},
                {mod,...},
                {...}|...]},
          {boot_rel,"start_clean"},
          {emu_name,"beam"},
          {relocatable,true},
          {profile,development},
          {incl_sys_files,[".*"]},
          {excl_sys_files,[]},
          {incl_app_files,[".*"]},
          {excl_app_files,[]},
          {incl_archive_dirs,[".*"]},
          {excl_archive_dirs,["^include$",[...]]},
          {archive_opts,[]},
          {app_type,permanent},
          {app_file,...},
          {...}]}}


7> reltool:get_config([{sys,[{profile, embedded}]}]).
{ok,{sys,[{profile,embedded},
          {incl_sys_filters,["^bin","^erts","^lib","^releases"]},
          {excl_sys_filters,["^bin/(erlc|dialyzer|typer)(|\\.exe)$",
                             "^erts.*/bin/(erlc|dialyzer|typer)(|\\.exe)$",
                             "^erts.*/bin/.*(debug|pdb)"]},
          {incl_app_filters,["^ebin","^include","^priv"]}]}}
8> reltool:get_config([{sys,[{profile, standalone}]}]).
{ok,{sys,[{profile,standalone},
          {incl_sys_filters,["^bin/(erl|epmd)(|\\.exe|\\.ini)$",
                             "^bin/start(|_clean).boot$","^erts.*/bin","^lib$"]},
          {excl_sys_filters,["^erts.*/bin/(erlc|dialyzer|typer)(|\\.exe)$",
                             "^erts.*/bin/(start|escript|to_erl|run_erl)(|\\.exe)$",
                             "^erts.*/bin/.*(debug|pdb)"]},
          {incl_app_filters,["^ebin","^priv"]},
          {excl_app_filters,["^ebin/.*\\.appup$"]}]}}
```

1.3.3 Generate release and script files

```
5> {ok, Server} = reltool:start_server([{config, {sys, [{boot_rel, "NAME"},
                                                        {rel, "NAME", "VSN",
                                                        [sasl]}]}}]).
{ok,<0.1288.0>}
6>  reltool:get_config(Server).
{ok,{sys,[{boot_rel,"NAME"},
          {rel,"NAME","VSN",[sasl]}]}}
7>  reltool:get_rel(Server, "NAME").
{ok,{release,{"NAME","VSN"},
             {erts,"5.7"},
             [{kernel,"2.13"},{stdlib,"1.16"},{sasl,"2.1.6"}]}}
8>  reltool:get_script(Server, "NAME").
{ok,{script,{"NAME","VSN"},
            [{preLoaded,[erl_prim_loader,erlang,init,otp_ring0,
                         prim_eval,prim_file,prim_inet,prim_zip,
                         zlib]},
             {progress,preloaded},
```

```
             {path,["$ROOT/lib/kernel-2.13/ebin",
                    "$ROOT/lib/stdlib-1.16/ebin"]},
             {primLoad,[error_handler]},
             {kernel_load_completed},
             {progress,kernel_load_completed},
             {path,["$ROOT/lib/kernel-2.13/ebin"]},
             {primLoad,[application,application_controller,
                        application_master,application_starter,auth,code,
                        code_server,disk_log,disk_log_1,disk_log_server,
                        disk_log_sup,dist_ac,dist_util,erl_boot_server|...]},
             {path,["$ROOT/lib/stdlib-1.16/ebin"]},
             {primLoad,[array,base64,beam_lib,c,calendar,dets,
                        dets_server,dets_sup,dets_utils,dets_v8,dets_v9,dict|...]},
             {path,["$ROOT/lib/sasl-2.1.6/ebin"]},
             {primLoad,[alarm_handler,erlsrv,format_lib_supp,misc_supp,
                        overload,rb,rb_format_supp,release_handler,
                        release_handler_1,sasl|...]},
             {progress,modules_loaded},
             {path,["$ROOT/lib/kernel-2.13/ebin",
                    "$ROOT/lib/stdlib-1.16/ebin","$ROOT/lib/sasl-2.1.6/ebin"]},
             {kernelProcess,heart,{heart,start,[]}},
             {kernelProcess,error_logger,{error_logger,start_link,[]}},
             {kernelProcess,application_controller,
                            {application_controller,start,[{...}]}},
             {progress,init_kernel_started},
             {apply,{application,load,[...]}},
             {apply,{application,load,...}},
             {progress,applications_loaded},
             {apply,{...}},
             {apply,...},
             {...}|...]}
9>  reltool:stop(Server).
ok
```

1.3.4 Create a target system

```
Erlang R13B02 (erts-5.7.3) [source] [64-bit] [smp:4:4] [rq:4]
                           [async-threads:0] [kernel-poll:false]

Eshell V5.7.3  (abort with ^G)
1> Config = {sys, [{escript,
                    "examples/display_args",
                    [{incl_cond, include}]},
     {app, inets, [{incl_cond, include}]},
     {app, mnesia, [{incl_cond, exclude}]},
     {app, ssl, [{incl_cond, exclude}]},
     {app, runtime_tools, [{incl_cond, exclude}]},
     {app, syntax_tools, [{incl_cond, exclude}]}]}.
{sys,[{escript,"examples/display_args",[{incl_cond,include}]},
      {app,inets,[{incl_cond,include}]},
      {app,mnesia,[{incl_cond,exclude}]},
      {app,ssl,[{incl_cond,exclude}]},
      {app,runtime_tools,[{incl_cond,exclude}]},
      {app,syntax_tools,[{incl_cond,exclude}]}]}



2> {ok, Spec} = reltool:get_target_spec([Config]).
{ok,[{create_dir,"releases",
                 [{write_file,"start_erl.data","5.7.3 1.0"},
                  {create_dir,"1.0",
```

```
                           [{write_file,"start_clean.rel",
                                        [37,37,32,114,101,108,32,103,101,110,101|...]},
                            {write_file,"start_clean.script",
                                        [37,37,32,115,99,114,105,112,116,32|...]},
                            {write_file,"start_clean.boot",
                                        <<131,104,3,100,0,6,115,99,114,...>>},
                            {write_file,"start_sasl.rel",
                                        [37,37,32,114,101,108,32,103,101,110,101|...]},
                            {write_file,"start_sasl.script",
                                        [37,37,32,115,99,114,105,112,116,32|...]},
                            {write_file,"start_sasl.boot",
                                        <<131,104,3,100,0,6,115,99,114,...>>}]}]},
    {create_dir,"bin",
                [{copy_file,"display_args.escript",
                            "/clearcase/otp/tools/reltool/examples/display_args"},
                 {copy_file,"display_args","erts-5.7.3/bin/escript"},
                 {copy_file,"start","erts-5.7.3/bin/start"},
                 {copy_file,"erl","erts-5.7.3/bin/dyn_erl"},
                 {copy_file,"epmd","erts-5.7.3/bin/epmd"},
                 {copy_file,"to_erl","erts-5.7.3/bin/to_erl"},
                 {copy_file,"run_erl","erts-5.7.3/bin/run_erl"},
                 {copy_file,"escript","erts-5.7.3/bin/escript"},
                 {copy_file,"erlc","erts-5.7.3/bin/erlc"},
                 {copy_file,"dialyzer","erts-5.7.3/bin/dialyzer"},
                 {copy_file,"typer","erts-5.7.3/bin/typer"},
                 {write_file,"start_clean.boot",
                             <<131,104,3,100,0,6,115,...>>},
                 {write_file,"start_sasl.boot",<<131,104,3,100,0,6,...>>},
                 {write_file,"start.boot",<<131,104,3,100,0,...>>}]},
    {create_dir,"misc",
                [{copy_file,"makewhatis"},{copy_file,"format_man_pages"}]},
    {copy_file,"Install"},
    {create_dir,"usr",
                [{create_dir,"lib",
                             [{copy_file,"liberts_r.a"},{copy_file,"liberts.a"}]},
                 {create_dir,"include",
                             [{copy_file,"erl_fixed_size_int_types.h"},
                              {copy_file,"erl_int_sizes_config.h"},
                              {copy_file,"erl_memory_trace_parser.h"},
                              {create_dir,"obsolete",[{copy_file,"driver.h"}]},
                              {copy_file,"driver_int.h"},
                              {copy_file,"erl_driver.h"}]}]},
    {create_dir,"erts-5.7.3",
                [{create_dir,"lib",
                             [{create_dir,"internal",
                                          [{copy_file,"liberts_internal_r.a"},
                                           {copy_file,"liberts_internal.a"},
                                           {copy_file,"libethread.a"},
                                           {copy_file,"README"}]},
                              {copy_file,"liberts_r.a"},
                              {copy_file,"liberts.a"}]},
                 {create_dir,"bin",
                             [{copy_file,"start"},
                              {copy_file,"erl","erts-5.7.3/bin/dyn_erl"},
                              {copy_file,"epmd"},
                              {copy_file,"to_erl"},
                              {copy_file,"run_erl"},
                              {copy_file,"escript"},
                              {copy_file,"erlc"},
                              {copy_file,"dialyzer"},
                              {copy_file,"typer"},
                              {copy_file,"erlexec"},
                              {copy_file,[...]},
                              {copy_file,...},
                              {...}|...]},
```

```
                {create_dir,"doc",[]},
                {create_dir,"man",[]},
                {create_dir,"include",
                           [{create_dir,"internal",
                                       [{create_dir,"tile",[{copy_file,...},{...}]},
                                        {create_dir,"sparc64",[{...}]},
                                        {create_dir,"sparc32",[...]},
                                        {create_dir,[...],...},
                                        {create_dir,...},
                                        {...}|...]},
                            {copy_file,"erl_fixed_size_int_types.h"},
                            {copy_file,"erl_int_sizes_config.h"},
                            {copy_file,"erl_memory_trace_parser.h"},
                            {copy_file,"driver_int.h"},
                            {copy_file,"erl_driver.h"}]},
                {create_dir,"src",[{copy_file,"setuid_socket_wrap.c"}]}]},
    {create_dir,"lib",
                [{archive,"compiler-4.6.3.ez",[],
                          [{create_dir,"compiler-4.6.3",
                                       [{create_dir,"ebin",
                                                   [{copy_file,"compiler.appup"},
                                                    {copy_file,[...]},
                                                    {copy_file,...},
                                                    {...}|...]},
                                        {create_dir,"src",
                                                   [{copy_file,[...]},
                                                    {copy_file,...},{...}|...]}]}]},
                 {archive,"crypto-1.6.1.ez",[],
                          [{create_dir,"crypto-1.6.1",
                                       [{create_dir,"ebin",
                                                   [{copy_file,[...]},
                                                    {copy_file,...},{...}|...]},
                                        {create_dir,"src",[{copy_file,...},{...}|...]}]}]},
                 {create_dir,"crypto-1.6.1",
                             [{create_dir,"priv",
                                          [{create_dir,"lib",[{copy_file,[...]}]},
                                           {create_dir,"obj",[{copy_file,...},{...}]}]}]},
                 {archive,"erts-5.7.3.ez",[],
                          [{create_dir,"erts-5.7.3",
                                       [{create_dir,"ebin",[{...}|...]},
                                        {create_dir,"src",[...]}]}]},
                 {archive,"hipe-3.7.3.ez",[],
                          [{create_dir,"hipe-3.7.3",
                                       [{create_dir,"util",[...]},
                                        {create_dir,[...],...},
                                        {create_dir,...},
                                        {...}|...]}]},
                 {archive,"kernel-2.13.3.ez",[],
                          [{create_dir,"kernel-2.13.3",
                                       [{create_dir,[...],...},{create_dir,...},{...}]}]},
                 {create_dir,"kernel-2.13.3",
                             [{create_dir,"include",
                                          [{copy_file,[...]},{copy_file,...},{...}]}]},
                 {archive,"stdlib-1.16.3.ez",[],
                          [{create_dir,"stdlib-1.16.3",[{...}|...]}]},
                 {create_dir,"stdlib-1.16.3",
                             [{create_dir,"include",[{...}|...]}]}]}]}



3> TargetDir = "my_target_dir".
"my_target_dir"
4> reltool:eval_target_spec(Spec, code:root_dir(), TargetDir).
{error,"/clearcase/otp/tools/reltool/my_target_dir: no such file or directory"}
5> file:make_dir("my_target_dir").
```

```
ok
6> reltool:eval_target_spec(Spec, code:root_dir(), TargetDir).
ok
7> file:list_dir(TargetDir).
{ok,["lib","erts-5.7.3","usr","Install","misc","bin","releases"]}
8> file:list_dir(filename:join([TargetDir,"lib"])).
{ok,["stdlib-1.16.3","stdlib-1.16.3.ez","kernel-2.13.3",
     "kernel-2.13.3.ez","hipe-3.7.3.ez","erts-5.7.3.ez",
     "crypto-1.6.1","crypto-1.6.1.ez","compiler-4.6.3.ez"]}
9> file:make_dir("yet_another_target_dir").
ok
10> reltool:create_target(Config, "yet_another_target_dir").
ok
```

2 Reference Manual

`Reltool` is a release management tool. It analyses a given Erlang/OTP installation and determines various dependencies between applications. The `graphical` frontend depicts the dependencies and enables interactive customization of a target system. The backend provides a `batch` interface for generation of customized target systems.

reltool

Erlang module

This is an interface module for the Reltool application.

Reltool is a release management tool. It analyses a given Erlang/OTP installation and determines various dependencies between applications. The `graphical` frontend depicts the dependencies and enables interactive customization of a target system. The backend provides a `batch` interface for generation of customized target systems.

The tool uses an installed Erlang/OTP system as input. `root_dir` is the root directory of the analysed system and it defaults to the system executing `reltool`. Applications may also be located outside `root_dir`. `lib_dirs` defines library directories where additional applications may reside and it defaults to the directories listed by the operating system environment variable `ERL_LIBS`. See the module `code` for more info.

An application directory `AppDir` under a library directory is recognized by the existence of an `AppDir/ebin` directory. If this does not exist, `reltool` will not consider `AppDir` at all when looking for applications.

It is recommended that application directories are named as the application, possibly followed by a dash and the version number. For example `myapp` or `myapp-1.1`.

Finally single modules and entire applications may be read from Escripts.

Some configuration parameters control the behavior of Reltool on system (`sys`) level. Others provide control on application (`app`) level and yet others are on module (`mod`) level. Module level parameters override application level parameters and application level parameters override system level parameters. Escript `escript` level parameters override system level parameters.

The following top level `options` are supported:

`config`

This is the main option and it controls the configuration of `reltool`. It can either be a `sys` tuple or a name of a `file` containing a sys tuple.

`trap_exit`

This option controls the error handling behavior of `reltool`. By default the window processes traps exit, but this behavior can altered by setting `trap_exit` to `false`.

`wx_debug`

This option controls the debug level of `wx`. As its name indicates it is only useful for debugging. See `wx:debug/1` for more info.

Besides the already mentioned source parameters `root_dir` and `lib_dirs`, the following system (`sys`) level options are supported:

`erts`

Erts specific configuration. See application level options below.

`escript`

Escript specific configuration. An escript has a mandatory file name and escript level options that are described below.

`app`

Application specific configuration. An application has a mandatory name and application level options that are described below.

`mod_cond`

This parameter controls the module inclusion policy. It defaults to `all` which means that if an application is included (either explicitly or implicitly) all modules in that application will be included. This implies that both modules that exist in the `ebin` directory of the application, as well as modules that are named in the `app` file will be included. If the parameter is set to `ebin`, both modules in the `ebin` directory and derived modules are included. If the parameter is set to `app`, both modules in the `app` file and derived modules are included. `derived` means that only modules that are used by other included modules are included. The `mod_cond` setting on system level is used as default for all applications.

`incl_cond`

This parameter controls the application and escript inclusion policy. It defaults to `derived` which means that the applications that do not have any explicit `incl_cond` setting, will only be included if any other (explicitly or implicitly included) application uses it. The value `include` implies that all applications and escripts that do not have any explicit `incl_cond` setting will be included. `exclude` implies that all applications and escripts that do not have any explicit `incl_cond` setting will be excluded.

`boot_rel`

A target system may have several releases but the one given as `boot_rel` will be used as default when the system is booting up.

`rel`

Release specific configuration. Each release maps to a `rel`, `script` and `boot` file. See the module `systools` for more info about the details. Each release has a name, a version and a set of applications with a few release specific parameters such as type and included applications.

`relocatable`

This parameter controls whether the `erl` executable in the target system should automatically determine where it is installed or if it should use a hardcoded path to the installation. In the latter case the target system must be installed with `reltool:install/2` before it can be used. If the system is relocatable, the file tree containing the target system can be moved to another location without re-installation. The default is `true`.

`profile`

The creation of the specification for a target system is performed in two steps. In the first step a complete specification is generated. It will likely contain much more files than you are interested in in your customized target system. In the second step the specification will be filtered according to your filters. There you have the ability to specify filters per application as well as system wide filters. You can also select a `profile` for your system. Depending on the `profile`, different default filters will be used. There are three different profiles to choose from: `development`, `embedded` and `standalone`. `development` is default. The parameters that are affected by the `profile` are: `incl_sys_filters`, `excl_sys_filters`, `incl_app_filters` and `excl_app_filters`.

`app_file`

This parameter controls the default handling of the `app` files when a target system is generated. It defaults to `keep` which means that `app` files are copied to the target system and their contents are kept as they are. `strip` means that a new `app` file is generated from the contents of the original `app` file where the non included modules are removed from the file. `all` does also imply that a new `app` file is generated from the contents of the original `app` file, with the difference that all included modules are added to the file. If the application does not have any `app` file a file will be created for `all` but not for `keep` and `strip`.

`debug_info`

The `debug_info` parameter controls whether the debug information in the beam file should be kept (`keep`) or stripped `strip` when the file is copied to the target system.

`excl_lib`

> Warning:
>
> This option is experimental.

If the `excl_lib` option is set to `otp_root` then reltool will not copy anything from the Erlang/OTP installation ($OTP_ROOT) into the target structure. The goal is to create a "slim" release which can be used together with an existing Erlang/OTP installation. The target structure will therefore only contain a `lib` directory with the applications that were found outside of $OTP_ROOT (typically your own applications), and a `releases` directory with the generated `.rel`, `.script` and `.boot` files.

When starting this release, three things must be specified:

Which `releases` directory to use
Tell the release handler to use the `releases` directory in our target structure instead of `$OTP_ROOT/releases`. This is done by setting the SASL environment variable `releases_dir`, either from the command line (`-sasl releases_dir <target-dir>/releases`) or in `sys.config`.

Which boot file to use
The default boot file is `$OTP_ROOT/bin/start`, but in this case we need to specify a boot file from our target structure, typically `<target-dir>/releases/<vsn>/<RelName>`. This is done with the `-boot` command line option to `erl`

The location of our applications
The generated .script (and .boot) file uses the environment variable `$RELTOOL_EXT_LIB` as prefix for the paths to all applications. The `-boot_var` option to `erl` can be used for specifying the value of this variable, typically `-boot_var RELTOOL_EXT_LIB <target-dir>/lib`.

Example:

```
erl -sasl releases_dir \"mytarget/releases\" -boot mytarget/releases/1.0/myrel\
 -boot_var RELTOOL_EXT_LIB mytarget/lib
```

`incl_sys_filters`

This parameter normally contains a list of regular expressions that controls which files in the system should be included. Each file in the target system must match at least one of the listed regular expressions in order to be included. Further the files may not match any filter in `excl_sys_filters` in order to be included. Which application files should be included is controlled with the parameters `incl_app_filters` and `excl_app_filters`. This parameter defaults to `[".*"]`.

`excl_sys_filters`

This parameter normally contains a list of regular expressions that controls which files in the system should not be included in the target system. In order to be included, a file must match some filter in `incl_sys_filters` but not any filter in `excl_sys_filters`. This parameter defaults to `[]`.

`incl_app_filters`

This parameter normally contains a list of regular expressions that controls which application specific files that should be included. Each file in the application must match at least one of the listed regular expressions in order to be included. Further the files may not match any filter in `excl_app_filters` in order to be included. This parameter defaults to `[".*"]`.

`excl_app_filters`

This parameter normally contains a list of regular expressions that controls which application specific files should not be included in the target system. In order to be included, a file must match some filter in `incl_app_filters` but not any filter in `excl_app_filters`. This parameter defaults to `[]`.

`incl_archive_filters`

This parameter normally contains a list of regular expressions that controls which top level directories in an application should be included in an archive file (as opposed to being included as a regular directory outside the archive). Each top directory in the application must match at least one of the listed regular expressions in order to be included. Further the files may not match any filter in `excl_app_filters` in order to be included. This parameter defaults to `[".*"]`.

`excl_archive_filters`

This parameter normally contains a list of regular expressions that controls which top level directories in an application should not be included in an archive file. In order to be included in the application archive, a top directory must match some filter in `incl_archive_filters` but not any filter in `excl_archive_filters`. This parameter defaults to `["^include$","^priv$"]`.

`archive_opts`

This parameter contains a list of options that are given to `zip:create/3` when application specific files are packaged into an archive. Only a subset of the options are supported. The most useful options in this context are the ones that control which types of files should be compressed. This parameter defaults to `[]`.

On application (`escript`) level, the following options are supported:

`incl_cond`

The value of this parameter overrides the parameter with the same name on system level.

On application (`app`) level, the following options are supported:

`vsn`

The version of the application. In an installed system there may exist several versions of an application. The `vsn` parameter controls which version of the application will be chosen.

This parameter is mutual exclusive with `lib_dir`. If `vsn` and `lib_dir` are both omitted, the latest version will be chosen.

Note that in order for reltool to sort application versions and thereby be able to select the latest, it is required that the version id for the application consits of integers and dots only, for example `1`, `2.0` or `3.17.1`.

`lib_dir`

The directory to read the application from. This parameter can be used to point out a specific location to fetch the application from. This is useful for instance if the parent directory for some reason is no good as a library directory on system level.

This parameter is mutual exclusive with `vsn`. If `vsn` and `lib_dir` are both omitted, the latest version will be chosen.

Note that in order for reltool to sort application versions and thereby be able to select the latest, it is required that the version id for the application consits of integers and dots only, for example `1`, `2.0` or `3.17.1`.

`mod`

Module specific configuration. A module has a mandatory name and module level options that are described below.

`mod_cond`

The value of this parameter overrides the parameter with the same name on system level.

`incl_cond`

The value of this parameter overrides the parameter with the same name on system level.

`app_file`

The value of this parameter overrides the parameter with the same name on system level.

`debug_info`

The value of this parameter overrides the parameter with the same name on system level.

`incl_app_filters`

The value of this parameter overrides the parameter with the same name on system level.

`excl_app_filters`

The value of this parameter overrides the parameter with the same name on system level.

`incl_archive_filters`

The value of this parameter overrides the parameter with the same name on system level.

`excl_archive_filters`

The value of this parameter overrides the parameter with the same name on system level.

`archive_opts`

The value of this parameter overrides the parameter with the same name on system level.

On module (`mod`) level, the following options are supported:

`incl_cond`

This parameter controls whether the module is included or not. By default the `mod_cond` parameter on application and system level will be used to control whether the module is included or not. The value of `incl_cond` overrides the module inclusion policy. `include` implies that the module is included, while `exclude` implies that the module is not included. `derived` implies that the module is included if it is used by any other included module.

`debug_info`

The value of this parameter overrides the parameter with the same name on application level.

DATA TYPES

```
options()           = [option()]
option()            = {config, config() | file()}
                    | {trap_exit, bool()}
                    | {wx_debug, term()}
config()            = {sys, [sys()]}
sys()               = {root_dir, root_dir()}
                    | {lib_dirs, [lib_dir()]}
                    | {profile, profile()}
                    | {erts, app()}
                    | {escript, escript_file(), [escript()]}
                    | {app, app_name(), [app()]}
                    | {mod_cond, mod_cond()}
                    | {incl_cond, incl_cond()}
                    | {boot_rel, boot_rel()}
```

```
                    | {rel, rel_name(), rel_vsn(), [rel_app()]}
                    | {relocatable, relocatable()}
                    | {app_file, app_file()}
                    | {debug_info, debug_info()}
                    | {incl_sys_filters, incl_sys_filters()}
                    | {excl_sys_filters, excl_sys_filters()}
                    | {incl_app_filters, incl_app_filters()}
                    | {excl_app_filters, excl_app_filters()}
                    | {incl_archive_filters, incl_archive_filters()}
                    | {excl_archive_filters, excl_archive_filters()}
                    | {archive_opts, [archive_opt()]}
app()               = {vsn, app_vsn()}
                    | {lib_dir, lib_dir()}
                    | {mod, mod_name(), [mod()]}
                    | {mod_cond, mod_cond()}
                    | {incl_cond, incl_cond()}
                    | {debug_info, debug_info()}
                    | {app_file, app_file()}
      | {excl_lib, excl_lib()}
                    | {incl_sys_filters, incl_sys_filters()}
                    | {excl_sys_filters, excl_sys_filters()}
                    | {incl_app_filters, incl_app_filters()}
                    | {excl_app_filters, excl_app_filters()}
                    | {incl_archive_filters, incl_archive_filters()}
                    | {excl_archive_filters, excl_archive_filters()}
                    | {archive_opts, [archive_opt()]}
mod()               = {incl_cond, incl_cond()}
                    | {debug_info, debug_info()}
rel_app()           = app_name()
                    | {app_name(), app_type()}
                    | {app_name(), [incl_app()]}
                    | {app_name(), app_type(), [incl_app()]}
app_name()          = atom()
app_type()          = permanent | transient | temporary | load | none
app_vsn()           = string()
archive_opt         = zip_create_opt()
boot_rel()          = rel_name()
app_file()          = keep | strip | all
debug_info()        = keep | strip
dir()               = string()
escript()           = {incl_cond, incl_cond()}
escript_file()      = file()
excl_app_filters()  = regexps()
excl_archive_filters() = regexps()
excl_lib()          = otp_root
excl_sys_filters()  = regexps()
file()              = string()
incl_app()          = app_name()
incl_app_filters()  = regexps()
incl_archive_filters() = regexps()
incl_cond()         = include | exclude | derived
incl_sys_filters()  = regexps()
lib_dir()           = dir()
mod_cond()          = all | app | ebin | derived | none
mod_name()          = atom()
profile()           = development | embedded | standalone
re_regexp()         = string()
reason()            = string()
regexps()           = [re_regexp()]
                    | {add, [re_regexp()]}
                    | {del, [re_regexp()]}
rel_file()          = term()
rel_name()          = string()
rel_vsn()           = string()
relocatable         = boolean()
```

```
root_dir()          = dir()
script_file()       = term()
server()            = server_pid() | options()
server_pid()        = pid()
target_dir()        = file()
window_pid()        = pid()
base_dir()          = dir()
base_file()         = file()
top_dir()           = file()
top_file()          = file()
target_spec()       = [target_spec()]
                    | {create_dir, base_dir(), [target_spec()]}
                    | {create_dir, base_dir(), top_dir(), [target_spec()]}
                    | {archive, base_file(), [archive_opt()], [target_spec()]}
                    | {copy_file, base_file()}
                    | {copy_file, base_file(), top_file()}
                    | {write_file, base_file(), iolist()}
                    | {strip_beam_file, base_file()}
```

Exports

`create_target(Server, TargetDir) -> ok | {error, Reason}`

Types:

`Server = server()`
`TargetDir = target_dir()`
`Reason = reason()`

Create a target system. Gives the same result as `{ok,TargetSpec}=reltool:get_target_spec(Server)` and `reltool:eval_target_spec(TargetSpec,RootDir,TargetDir)`.

`eval_target_spec(TargetSpec, RootDir, TargetDir) -> ok | {error, Reason}`

Types:

`TargetSpec = target_spec()`
`RootDir = root_dir()`
`TargetDir = target_dir()`
`Reason = reason()`

Create the actual target system from a specification generated by `reltool:get_target_spec/1`. The creation of the specification for a target system is performed in two steps. In the first step a complete specification will be generated. It will likely contain much more files than you are interested in in your target system. In the second step the specification will be filtered according to your filters. There you have the ability to specify filters per application as well as system wide filters. You can also select a `profile` for your system. Depending on the `profile`, different default filters will be used.

The top directories `bin`, `releases` and `lib` are treated differently from other files. All other files are by default copied to the target system. The `releases` directory contains generated `rel`, `script`, and `boot` files. The `lib` directory contains the applications. Which applications are included and if they should be customized (archived, stripped from debug info etc.) is specified with various configuration parameters. The files in the `bin` directory are copied from the `erts-vsn/bin` directory, but only those files that were originally included in the `bin` directory of the source system.

If the configuration parameter `relocatable` was set to `true` there is no need to install the target system with `reltool:install/2` before it can be started. In that case the file tree containing the target system can be moved without re-installation.

In most cases, the `RootDir` parameter should be set to the same as the `root_dir` configuration parameter used in the call to `reltool:get_target_spec/1` (or `code:root_dir()` if the configuration parameter is not set). In some cases it might be useful to evaluate the same target specification towards different root directories. This should, however, be used with great care as it requires equivalent file structures under all roots.

`get_config(Server) -> {ok, Config} | {error, Reason}`

Types:

```
Server = server()
Config = config()
Reason = reason()
```

Get reltool configuration. Shorthand for `reltool:get_config(Server,false,false)`.

`get_config(Server, InclDefaults, InclDerived) -> {ok, Config} | {error, Reason}`

Types:

```
Server = server()
InclDefaults = incl_defaults()
InclDerived = incl_derived()
Config = config()
Reason = reason()
```

Get reltool configuration. Normally, only the explicit configuration parameters with values that differ from their defaults are interesting. But the builtin default values can be returned by setting `InclDefaults` to `true`. The derived configuration can be returned by setting `InclDerived` to `true`.

`get_rel(Server, Relname) -> {ok, RelFile} | {error, Reason}`

Types:

```
Server = server()
RelName = rel_name()
RelFile = rel_file()
Reason = reason()
```

Get contents of a release file. See `rel(4)` for more details.

`get_script(Server, Relname) -> {ok, ScriptFile | {error, Reason}`

Types:

```
Server = server()
RelName = rel_name()
ScriptFile = script_file()
Reason = reason()
```

Get contents of a boot script file. See `script(4)` for more details.

`get_status(Server) -> {ok, [Warning]} | {error, Reason}`

Types:

```
Server = server()
Warning = string()
```

Reason = reason()

Get status about the configuration

get_server(WindowPid) -> {ok, ServerPid} | {error, Reason}

Types:

WindowPid = window_pid()
ServerPid = server_pid()
Reason = reason()

Return the process identifier of the server process.

get_target_spec(Server) -> {ok, TargetSpec} | {error, Reason}

Types:

Server = server()
TargetSpec = target_spec()
Reason = reason()

Return a specification of the target system. The actual target system can be created with `reltool:eval_target_spec/3`.

install(RelName, TargetDir) -> ok | {error, Reason}

Types:

RelName = rel_name()
TargetDir = target_dir()
Reason = reason()

Install a created target system

start() -> {ok, WindowPid} | {error, Reason}

Types:

WindowPid = window_pid()
Reason = reason()

Start a main window process with default options

start(Options) -> {ok, WindowPid} | {error, Reason}

Types:

Options = options()
WindowPid = window_pid()
Reason = reason()

Start a main window process with options

start_link(Options) -> {ok, WindowPid} | {error, Reason}

Types:

Options = options()
WindowPid = window_pid()
Reason = reason()

Start a main window process with options. The process is linked.

start_server(Options) -> {ok, ServerPid} | {error, Reason}

Types:

Options = options()
ServerPid = server_pid()
Reason = reason()

Start a server process with options. The server process identity can be given as an argument to several other functions in the API.

stop(Pid) -> ok | {error, Reason}

Types:

Pid = server_pid() | window_pid()()
Reason = reason()

Stop a server or window process