

---

# Rivendell Macro Language

Fred Gleason

## Table of Contents

Scope .....	2
Protocol .....	2
Command Delivery .....	2
Command Reply .....	2
Specifying Color .....	3
Binary Data .....	3
Commands .....	3
<b>Add Next [PX]</b> .....	3
<b>Binary Serial Out [BO]</b> .....	4
<b>Clear Serial Trap [SC]</b> .....	4
<b>Command Send [CC]</b> .....	4
<b>Connect Jack Ports [JC]</b> .....	4
<b>Console Label [CL]</b> .....	5
<b>Copy Cut [CP]</b> .....	5
<b>Cut Event [CE]</b> .....	5
<b>Database Backup [DB]</b> .....	5
<b>Disconnect Jack Ports [JD]</b> .....	5
<b>Duck Panel Button [PD]</b> .....	6
<b>Execute Breakaway [DX]</b> .....	6
<b>Execute Cart [EX]</b> .....	6
<b>Fire Salvo [FS]</b> .....	6
<b>GPI Enable [GE]</b> .....	7
<b>GPI Enable [GE]</b> (old format, deprecated) .....	7
<b>GPI Set [GI]</b> .....	7
<b>GPI Set [GI]</b> (old format, deprecated) .....	7
<b>GPO Set [GO]</b> .....	8
<b>GPO Set [GO]</b> (old format, deprecated) .....	8
<b>Insert Serial Trap [SI]</b> .....	8
<b>Label Panel [PC]</b> .....	8
<b>Load Log [LL]</b> .....	9
<b>Load Panel [PE]</b> .....	9
<b>Load Slot [DL]</b> .....	9
<b>Login [LO]</b> .....	10
<b>Macro Timer [MT]</b> .....	10
<b>Make Next [MN]</b> .....	10
<b>Message Box [MB]</b> .....	10
<b>Pause Panel [PU]</b> .....	11
<b>Play Panel [PP]</b> .....	11
<b>Play Slot [DP]</b> .....	11
<b>Refresh Log [RL]</b> .....	11
<b>Run Shell Command [RN]</b> .....	12
<b>Select Widget [PW]</b> .....	12
<b>Serial Out [SO]</b> .....	12
<b>Serial Reload [SY]</b> .....	12
<b>Set Color Label [LC]</b> .....	12

Set Default Now & Next Cart [SN] .....	13
Set Display [SD] .....	13
Set Duck Level [MD] .....	14
Set Label [LB] .....	14
Set Mode [PM] .....	14
Sleep [SP] .....	14
Start [PL] .....	15
Start Button [PB] .....	15
Start Next [PN] .....	15
Start Record Deck [RS] .....	15
Stop [PS] .....	16
Stop Panel [PT] .....	16
Stop Record Deck [RR] .....	16
Stop Slot [DS] .....	16
Switch Add [SA] .....	17
Switch Add With Gain [SG] .....	17
Switch Crosspoint Gain [SX] .....	17
Switch Level [SL] .....	17
Switch Reload [SZ] .....	17
Switch Remove [SR] .....	18
Switch Take [ST] .....	18
Toggle On Air Flag [TA] .....	18
UDP Out [UO] .....	18

## Scope

Rivendell Macro Language (RML) is a system of commands formulated to specify actions to be taken by and within the Rivendell Radio Automation system.

## Protocol

The Normal form of a command takes the following syntax:

*cmd* [*arg*] [. . .]!

*cmd*                      A command mnemonic consisting of two alpha-numeric characters indicating the action to be performed.

*arg*                      Zero or more arguments, delimited by spaces or, if the last argument, by ! (see below)

!                          The ASCII character 33, indicating the end of the command sequence.

## Command Delivery

To be executed by a Rivendell system, an appropriate command should be delivered to UDP ports 5858 or 5859 on said system, using the SOCK\_DGRAM connectionless protocol. Commands delivered to port 5858 may receive a reply back to the originating IP address at UDP socket 5860 to indicated command success/failure, while commanders delivered to 5859 will be processed, but no reply made.

## Command Reply

If delivered to UDP port 5858, each command may receive a reply, formulated as follows:

*rpy* +|-

*rpy*                      The full command string (including arguments) originally received

  +                      ACK response, indicates success of the command.

  -                      NAK response, indicates failure of the command.

## Specifying Color

In places where a color can be specified as a macro argument, the following predefined values are valid:

**white**  
**red**  
**green**  
**blue**  
**cyan**  
**magenta**  
**yellow**  
**gray**  
**lightGray**  
**black**  
**darkRed**  
**darkGreen**  
**darkBlue**  
**darkCyan**  
**darkMagenta**  
**darkYellow**  
**darkGray**

## Binary Data

Certain commands (e.g. **SO** and **UO**) have the ability to transmit arbitrary binary data in addition to textual characters. Such arbitrary binary data can be specified by means of escape codes as follows:

*%hexcode*

*hexcode*                Two digit numeric value of the byte to send in hexadecimal.

For example, the sequence 'TEST' followed by a carriage return/newline could be specified as:

**TEST%0D%0A**

## Commands

### Add Next [PX]

Module        **rdairplay**(1)

Mnemonic     **PX**

Insert a cart in a log in the next to play position.

**PX *mach cart*!**

Insert cart *cart*> in the next to play position on log machine *mach*.

## Binary Serial Out [BO]

Module      **ripd(8)**

Mnemonic    **BO**

Output a string of binary codes.

**BO *portnum hexcode* ..!**

Output a string of binary codes represented by *hexcode* .. on serial port *portnum*!

## Clear Serial Trap [SC]

Module      **ripd(8)**

Mnemonic    **SC**

Clear a serial trap.

**SC *portnum cart string*!**

**SC *portnum cart*!**

**SC *portnum*!**

Clear a serial trap. The three argument form will clear all traps on *port* that reference *cart* and *string*. The two argument form will clear all traps on *port* that reference *cart*, while the one argument form will clear all traps on *port*.

## Command Send [CC]

Module      **ripd(8)**

Mnemonic    **CC**

Send RML to a remote host.

**CC *dest [:udpport] rml*!**

Send the RML command *rml* to *dest*. *dest* may be either a Rivendell host name or an IP address. A UDP port value may be optionally specified as *udpport* (default value 5859).

## Connect Jack Ports [JC]

Module      **ripd(8)**

Mnemonic    **JC**

Connect a JACK input port to an output port.

**JC *output input*!**

Attempt to connect JACK input port *input* to *output*.

## Console Label [CL]

Module      **ripd(8)**

Mnemonic    **CL**

Set the label on a control surface.

**CL *matrix surface chan label!***

On matrix *matrix*, set the input *chan* module label of control surface *surface* to *label*.

## Copy Cut [CP]

Module      **rdcatchd(8)**

Mnemonic    **CP**

Copy cut audio and metadata.

**CP *srccart srccut dstcart dstcut!***

Copy the audio and metadata from cut number *srccut* in cart *srccart* to cut number *dstcut* in cart *dstcart*. The destination cart/cut must already exist.

## Cut Event [CE]

Module      **rdcatchd(8)**

Mnemonic    **CE**

Place a cut event at the current location of an active RDCatch recording.

**CE *decknum eventnum!***

Place a cut event with event number *eventnum* at the current position of the active recording in RDCatch Deck *decknum*.

## Database Backup [DB]

Module      **ripd(8)**

Mnemonic    **DB**

Backup the Rivendell database.

**DB *filename!***

Create a backup copy of the active Rivendell database in *filename*.

## Disconnect Jack Ports [JD]

Module      **ripd(8)**

Mnemonic    **JD**

Disconnect a JACK input port from an output port.

**JD *output input*!**

Attempt to disconnect JACK input port *input* from *output*.

## Duck Panel Button [PD]

Module **rdairplay(1)**

Mnemonic **PD**

Set the duck level of a SoundPanel button in RDAirPlay.

**PD *panel col row level fade [mport]*!**

Set the duck level of button at *column, row* of panel *panel*. Panel: **S1...S50** or **U1...U50**; **C** for *panel* means currently selected panel. If *col* and/or *row* is **0**: duck all buttons in *col* or *row*. If playing, the audio fades to *level* (in dbFS) in *fade* msec. If specified, only stop carts playing on panel port *mport*. The duck level will be set to 0 after the playback is finished, or the button is stopped or reset.

## Execute Breakaway [DX]

Module **rdcartslots(1)**

Mnemonic **DX**

Execute a breakaway.

**DX *slotlen*!**

Execute a breakaway of length *len* (in mS) using the autofill carts configured for the service currently loaded in the *slot* slot. The slot must be in Breakaway mode. If a breakaway is currently executing, it will immediately be aborted and a new one started. Passing a '0' for *len* will cause any currently executing breakaway immediately to be aborted and input audio restored.

## Execute Cart [EX]

Module **rdcatchd(8)**

Mnemonic **EX**

Execute a macro cart.

**EX *cartnum*!**

Execute macro cart *cartnum* on the local host.

## Fire Salvo [FS]

Module **ripd(8)**

Mnemonic **FS**

Execute a switcher salvo.

**FS *matrix salvo*!**

Fire salvo number *salvo* on matrix *matrix*.

## GPI Enable [GE]

Module      **ripd(8)**

Mnemonic    **GE**

Enable/disable a GPI line.

**GE *matrix type gpinum state*!**

Enable or disable the GPI line of type *type* indicated by *gpinum* on matrix *matrix*. Possible types are:

- I**    Input
- O**    Output

## GPI Enable [GE] (old format, deprecated)

Module      **ripd(8)**

Mnemonic    **GE**

Enable/disable a GPI line.

**GE *matrix gpinum state*!**

Enable or disable the GPI line indicated by *gpinum* on matrix *matrix*.

## GPI Set [GI]

Module      **ripd(8)**

Mnemonic    **GI**

Set action in response to a GPIO event.

**GI *matrix type gpinum state cart*!**

Execute the macro *cart cart* upon transition of the GPIO of type *type* and line *gpinum* on matrix *matrix* to *state*. Possible types are:

- I**    Input
- O**    Output

## GPI Set [GI] (old format, deprecated)

Module      **ripd(8)**

Mnemonic    **GI**

Set action in response to a GPIO event.

**GI *matrix gpinum state cart!***

Execute the macro *cart* upon transition of the GPIO of line *gpinum* on matrix *matrix* to *state*.

## GPO Set [GO]

Module **ripd(8)**

Mnemonic **GO**

Set the state of a GPIO line.

**GO *matrix type gponum state length!***

Command GPIO line *gponum* of type *type* on matrix *matrix* to *state* for *length* mS. A length of 0 indicates to hold the state indefinitely. Possible types are:

**I** Input

**O** Output

Possible states are:

**0** OFF

**1** ON

**-1** Passthrough hardware input (valid only for Input type)

## GPO Set [GO] (old format, deprecated)

Module **ripd(8)**

Mnemonic **GO**

Set the state of a GPIO line.

**GO *matrix gponum state length!***

Command GPIO line *gponum* on matrix *matrix* to *state* for *length* mS. A length of 0 indicates to hold the state indefinitely.

## Insert Serial Trap [SI]

Module **ripd(8)**

Mnemonic **SI**

Insert a serial trap to detect a character sequence.

**SI *portnum cart string!***

Insert a serial trap to execute the macro *cart* upon receipt of *string* on serial port *port*.

## Label Panel [PC]

Module **rdairplay(1)**



Mnemonic    **PC**

Set the label of a SoundPanel button in RDAirPlay.

**PC *panel col row label color!***

Set the button at *col, row* of panel *panel* to have a text label of *label* and a background color of *color*. Panel: **S1...S50** or **U1...U50**; **C** for *panel* means currently selected panel.

## Load Log [LL]

Module        **rdairplay(1)**

Mnemonic    **LL**

Load a log into RDAirPlay.

**LL *mach [logname] [startline]!***

Load the log *logname* in log machine *mach*. After loading, start the log at line *startline* if it is  $\geq 0$ . If *startline* is -2, the log will be started at the first event if that event does not have a 'stop' transition. Default *startline* = -1. If no *log* is specified, the machine's current log is unloaded.

## Load Panel [PE]

Module        **rdairplay(1)**

Mnemonic    **PE**

Load a cart into a SoundPanel button in RDAirPlay.

**PE *panel col row cart!***

Load cart *cart* into the button at *col, row* of panel *panel*. Panel: **S1...S50** or **U1...U50**; **C** for *panel* means currently selected panel.

## Load Slot [DL]

Module        **rdcartslots(1)**

Mnemonic    **DL**

Load an RDCartSlots slot.

This command has two syntaxes, depending upon the mode of the slot.

Cart Deck Mode syntax:

**DL *slot cart!***

Load the cart *cart* into the *slot* slot. Passing a '0' for *cart* will cause the slot to be unloaded. This command will be ignored if the slot is currently playing.

Breakaway Mode syntax:

**DL *slot svcname!***

Set the *slot* slot to use service *svcname*. Omitting the *svcname* argument will cause the slot to be unloaded. This command will be ignored if the slot is currently executing a breakaway.

## Login [LO]

Module      **ripd(8)**

Mnemonic    **LO**

Change the active Rivendell user context.

**LO *user password*!**

Set the current Rivendell user to *user*. If no arguments are supplied, log out the station --i.e. revert to the default user.

## Macro Timer [MT]

Module      **ripd(8)**

Mnemonic    **MT**

Timer for running macro carts.

**MT *timernum timeout cart*!**

Set the macro timer *timernum* to execute macro cart *cart* in *timeout* milliseconds. Setting *timeout* to 0 disables the timer. Sixteen macro timers (numbered 1-16) are available on each Rivendell host.

## Make Next [MN]

Module      **rdairplay(1)**

Mnemonic    **MN**

Load changes to a currently loaded log in RDAirPlay.

**RL *mach line*!**

Set the next event for log machine *mach* to line *line*>.

## Message Box [MB]

Module      **ripd(8)**

Mnemonic    **MB**

Display a popup message box on a host display.

**MB *display severity msg*!**

Display the text *msg* in a popup window on X display *display*, with an icon to indicate *severity*. Valid values of *severity* are:

- 1 Information

2 Warning

3 Critical

## Pause Panel [PU]

Module **rdairplay**(1)

Mnemonic **PU**

Pause a SoundPanel button in RDAirPlay.

**PU** *panel col row [mport]!*

Pause the button at *col*, *row* of panel *panel*. Panel: **S1...S50** or **U1...U50**; **C** for *panel* means currently selected panel. If *col* and/or *row* is **0**: pause all playing buttons in *col* or *row*. If specified, start the playout on panel port *mport*.

## Play Panel [PP]

Module **rdairplay**(1)

Mnemonic **PP**

Load a cart into a SoundPanel button in RDAirPlay.

**PP** *panel col row [mport] [0|1]!*

Play the button at *col*, *row* of panel *panel*. Panel: **S1...S50** or **U1...U50**; **C** for *panel* means currently selected panel. If *col* and/or *row* is **0**: Start the first loaded button that is not active. If specified, start the playout on panel port *mport*. The started panel will stay active when finished, if the 5th argument is **1**.

## Play Slot [DP]

Module **rdcartslots**(1)

Mnemonic **DP**

Play an RDCartSlots slot.

**DP** *slot!*

Play the cart currently loaded in the *slot* slot. The slot must be in Cart Deck mode. This command will be ignored if the slot is unloaded or already playing.

## Refresh Log [RL]

Module **rdairplay**(1)

Mnemonic **RL**

Load changes to a currently loaded log in RDAirPlay.

**RL** *mach!*

Refresh the log currently loaded in log machine *mach*.

## Run Shell Command [RN]

Module      **ripd(8)**

Mnemonic    **RN**

Run a shell command.

**RN *cmd*!**

Run the shell command *cmd*.

## Select Widget [PW]

Module      **rdairplay(1)**

Mnemonic    **PW**

Select right-hand widget in RDAirPlay.

**PW *mach*!**

Select right-hand widget to log-machine *mach* or **0** for sound panel.

## Serial Out [SO]

Module      **ripd(8)**

Mnemonic    **SO**

Output a string on a serial port.

**SO *portnum data*!**

Output *data* on serial port *portnum*. *data* can consist of arbitrary binary data as well as textual characters (see **Binary Data** above).

## Serial Reload [SY]

Module      **ripd(8)**

Mnemonic    **SY**

Reload the configuration for a serial port.

**SY *portnum*!**

Reload the configuration for serial port *portnum*. Normally, this should only be issued by RDAdmin following a configuration change.

## Set Color Label [LC]

Module      **rdairplay(1)**

Mnemonic    **LC**

Display a color message in the label widget on RDAirPlay.

**LC *color string*!**

Display *string* in color *color* in the message widget.

## Set Default Now & Next Cart [SN]

Module      **rdairplay(1)**

Mnemonic    **SN**

Set the default Now & Next cart.

**SN now|next *mach cart*!**

Set the default Now & Next cart for log *mach* to *cart*.

## Set Display [SD]

Module      **ripd(8)**

Mnemonic    **SD**

Set a console display.

**SD *matrix display line col attr label*!**

On matrix *matrix*, set the console display *display* to *label*, starting at position *line*, *col* and using message attributes *attr*.

The message attributes value is constructed as follows:

Bit 7            Display mode

Bits 6,5        Video attribute

Bit 2,1,0       Message Text Color

Display Mode:

0    Overwrite text

1    Insert text

Video Attribute:

00   Normal

01   Flash

02   Reverse

Text Color:

000   White

001   Red

010 Yellow

011 Green

100 Cyan

101 Magenta

## Set Duck Level [MD]

Module **rdairplay**(1)

Mnemonic **MD**

Set duck level for an RDAirPlay log machine.

**MD *mach level fade* [*mport*]!**

Set the duck level of *mach*, or **0** for all log machines. If playing, the audio fades to *level* (in dbFS) in *fade* msecs. If specified, only affect carts playing on machine port *mport*. Loading or clearing a log will set the duck level back to 0.

## Set Label [LB]

Module **rdairplay**(1)

Mnemonic **LB**

Display a message in the label widget on RDAirPlay.

**LB *string*!**

Display *string*> in the message widget.

## Set Mode [PM]

Module **rdairplay**(1)

Mnemonic **PM**

Set the mode of an RDAirPlay log machine.

**PM *mode* [*mach*]!**

Set log machine *mach* to mode *mode*. If mode is not given or the Mode Control Style is set to 'Unified', then all log machines are set to *mode*. Valid values for *mode* are:

1 LiveAssist

2 Auto

3 Manual

## Sleep [SP]

Module **ripd**(8)

Mnemonic    **SP**

Pause for specified time.

**SP *msecs*!**

Wait for *msecs* milliseconds.

## Start [PL]

Module        **rdairplay(1)**

Mnemonic    **PL**

Start a log at a specified line.

**PL *mach line*!**

Start log machine *mach* at line *line* if stopped, otherwise do nothing.

## Start Button [PB]

Module        **rdairplay(1)**

Mnemonic    **PB**

Push an RDAirPlay Start button.

**PB *button*!**

Push button *button*.

## Start Next [PN]

Module        **rdairplay(1)**

Mnemonic    **PN**

Start the next event in a log.

**PN *mach* [*mport*] [*skip*]!**

Start log machine *mach*> if stopped, or start next event if already running. If specified, start the playout on machine port *mport*>. If *skip* is supplied, equal to '1' and the log machine is in Manual or Live Assist mode, then any intervening meta-events in log between the current 'next' event and the next cart will be skipped over.

## Start Record Deck [RS]

Module        **rdcatchd(8)**

Mnemonic    **RS**

Start an RDCatch Recording

**RS *decknum cartnum cutnum maxlen*!**

Start recording to cut *cutnum* of cart *cartnum*, using RDCatch record deck *decknum* for a maximum time of *maxlen* mS. The record parameters used (format, sample rate, channels, etc) will be those configured for the selected deck in RAdmin->ManageHosts->RDCatch. The selected cart and cut must already exist. Any audio previously residing in the selected cart and cut will be overwritten.

## Stop [PS]

Module **rdairplay(1)**

Mnemonic **PS**

Stop a log in RDAirPlay.

**PS mach|0 [fade] [mport]!**

Stop log machine *mach*, or **0** for all log machines. If specified, only stop carts playing on machine port *mport*. If specified, fade out *fade* msecs.

## Stop Panel [PT]

Module **rdairplay(1)**

Mnemonic **PT**

Stop a SoundPanel button in RDAirPlay.

**PT panel col row [mport] [0|1] fade!**

Stop the button at *col*, *row* of panel *panel*. Panel: **S1...S50** or **U1...U50**; **C** for *panel* means currently selected panel. If *col* and/or *row* is **0**: stop all playing buttons in *col* or *row*. If specified, start the ployout on panel port *mport*. The stopped panel will stay active when finished, if the 5th argument is **1**.

## Stop Record Deck [RR]

Module **rdcatchd(8)**

Mnemonic **RR**

Stop an RDCatch Recording

**RR decknum!**

Stop any active recording on RDCatch deck *decknum*.

## Stop Slot [DS]

Module **rdcartslots(1)**

Mnemonic **DS**

Stop an RDCartSlots slot.

**DS slot!**

Stop the cart currently loaded in the *slot* slot. The slot must be in Cart Deck mode. This command will be ignored if the slot is unloaded or already playing.



## Switch Add [SA]

Module      **ripcd(8)**

Mnemonic    **SA**

Add an input to an output.

***SA matrix input output!***

Command switch matrix number *matrix* to add input number *input* to output number *output*. Unlike **SWITCH TAKE**, this command leaves any other previously assigned inputs unchanged.

## Switch Add With Gain [SG]

Module      **ripcd(8)**

Mnemonic    **SG**

Add an input to an output while specifying the crosspoint gain.

***SG matrix input output gain!***

Command switch matrix number *matrix* to add input number *input* to output number *output* at gain *gain*. The gain is specified in 1/10 of a dB, with 0 = unity gain. Unlike **SWITCH TAKE**, this command leaves any other previously assigned inputs unchanged.

## Switch Crosspoint Gain [SX]

Module      **ripcd(8)**

Mnemonic    **SX**

Set crosspoint gain.

***SX matrix input output level!***

Command switch matrix number *matrix* to adjust the gain of the crosspoint connecting input *input* to output *output* to *level* dB.

## Switch Level [SL]

Module      **ripcd(8)**

Mnemonic    **SL**

Set input gain.

***SL matrix input level!***

Command switch matrix number *matrix* to adjust the gain of input number *input* to *level* dB.

## Switch Reload [SZ]

Module      **ripcd(8)**

Mnemonic    **SZ**

Reload the configuration for a switch matrix.

**SZ *matrix*!**

Reload the configuration for switch matrix *matrix*. Normally, this should only be issued by RDAdmin following a configuration change.

## Switch Remove [SR]

Module        **ripd(8)**

Mnemonic    **SR**

Remove an input from an output.

**SR *matrix input output*!**

Command switch matrix number *matrix* to remove input number *input* from output number *output*. Unlike **SWITCH TAKE**, this command leaves any other previously assigned inputs unchanged.

## Switch Take [ST]

Module        **ripd(8)**

Mnemonic    **ST**

Exclusively route an input to an output.

**ST *matrix input output*!**

Command switch matrix number *matrix* to take input number *input* to output number *output*. "Take" in this context implies removing any previously assigned inputs from the referenced output.

## Toggle On Air Flag [TA]

Module        **ripd(8)**

Mnemonic    **TA**

Set the state of the On Air flag.

**TA 0|1!**

Set the On-Air flag to ON [1] or OFF [0].

## UDP Out [UO]

Module        **ripd(8)**

Mnemonic    **UO**

Send data to a UDP port.

**UO *ipaddr udpport data*!**

Send *data* in a UDP packet to port *udpport* at *ipaddr*. *data* can consist of arbitrary binary data as well as textual characters (see **Binary Data** above).