

---

## Name

rd\_savelog — Rivendell Save Log C Library Function

## Synopsis

```
#include <rivwebcapi/rd_savelog.h>

int    RD_SaveLog(*hdrvals,    *linevals,    linevals_quan,    hostname[],
username[], passwd[], ticket[], log_name[], user_agent[]);

struct save_loghdr_values *hdrvals;
struct save_logline_values *linevals;
unsigned linevals_quan;
const char hostname[];
const char username[];
const char passwd[];
const char ticket[];
const char log_name[];
const char user_agent[];
```

## Description

**RD\_SaveLog** is the function to use to save log information (both headers and lines) to an existing Rivendell Database.

This function saves the specified log to the Rivendell database on hostname.

**Table 1. RD\_SaveLog function call fields**

FIELD NAME	FIELD TYPE	MEANING	REMARKS
hdrvals	Pointer to save_loghdr_values structure	Log header data	Mandatory
linevals	Pointer to save_logline_values structure	Log lines data	Mandatory
linevals_quan	Unsigned Integer	Number of lines in the linevals argument	Mandatory
hostname	Character Array	Name Of Rivendell DB Host	Mandatory
username	Character Array	Rivendell User Name	Mandatory When NO Ticket Provided
passwd	Character Array	Rivendell User Password	Mandatory When NO Ticket Provided
ticket	Character Array	Rivendell Authentication Ticket	Mandatory When NO User/Password Pair Provided.
log_name	Character Array	Name to assign to new log	Mandatory

FIELD NAME	FIELD TYPE	MEANING	REMARKS
user_agent	Character Array	User Agent Value put into HTTP request	Optional (default is Rivendell-C-API/x.x.x)

Header data for the log is sent in a 'save\_loghdr\_values' struct, as follows:

```
struct save_loghdr_values {
    char loghdr_service[11];
    char loghdr_description[65];
    int loghdr_autorefresh;
    struct tm loghdr_purge_date;
    struct tm loghdr_start_date;
    struct tm loghdr_end_date;
};
```

Each line in the log is sent in a 'save\_logline\_values' struct, as follows:

```
struct save_logline_values {
    int logline_id;
    int logline_type;
    int logline_source;
    unsigned logline_cart_number;
    int logline_starttime;
    int logline_gracetime;
    int logline_time_type;
    int logline_transition_type;
    int logline_start_point_log;
    int logline_end_point_log;
    int logline_segue_start_point_log;
    int logline_segue_end_point_log;
    int logline_fadeup_point_log;
    int logline_fadedown_point_log;
    int logline_duckup_gain;
    int logline_duckdown_gain;
    char logline_marker_comment[1021];
    char logline_marker_label[257];
    char logline_origin_user[1021];
    struct tm logline_origin_datetime;
    int logline_event_length;
    char logline_link_event_name[257];
    struct tm logline_link_starttime;
    int logline_link_length;
    int logline_link_start_slop;
    int logline_link_end_slop;
    int logline_link_id;
    int logline_link_embedded;
    struct tm logline_ext_starttime;
    int logline_ext_length;
    char logline_ext_cart_name[129];
    char logline_ext_data[129];
    char logline_ext_event_id[129];
};
```

```
char logline_ext_annc_type[33];  
};
```

## RETURN VALUE

On success, zero is returned.

If a server error occurs a -1 is returned. If a client error occurs a specific error number is returned.

## ERRORS

400 Invalid/Missing Parameter.

403 User Authentication Error.

404 User Permission Error.

404 Service or Log Does Not Exist.

500 Server error.

nnn Unknown Error Occurred.