RSJ CD Writer Application Programming Interface

For Windows NT / 2000
And Windows 9x / Me

http://www.rsj.de

Notes

RSJ CD Writer provides an application programming interface (API) to allow other applications to use the functionality of the RSJ CD Writer File System.

Note: The API has been created with the Microsoft Visual C++ 4.2/6.0 compiler. It has been tested with Visual Basic 5.0. Other combinations might work as well, but this has not been verified. Please report problems, comments or suggestions to support@rsj.de

The functionality of this API does not include track copy functions.

You and your clients need RSJ CD Writer for Win95 version 1.27 or above or RSJ CD Writer for Windows NT/2000 for this API to work!

Please note the Conditions Of Use in the *RSJ CD Writer Owner's Manual*.

Contents

General Information

The RSJ CD Writer API contains the following files:

CDWACC.H	The header file „cdwacc.h“ has to be included in each C/C++ module that uses the CD Writer API. It defines the types and constants that are used to communicate with the file system. It also provides latest information about the API.
CDWAPI.TXT	This file contains definitions and declarations needed to use this API from Visual Basic Applications. It can be inserted in any Visual Basic project as a code module. This file is part of the Visual Basic programming sample available on our web site (download section).
CDWACC.LIB	This is the C import library of the API.
CDWACC.DLL	The API module itself.
CDWAPI.PDF	This documentation in PDF (Adobe Acrobat) format.

Important notes:

- The RSJ CD Writer API is implemented in cdwacc.dll, which is part of the normal distribution.
- Do not redistribute any files which are part of RSJ CD Writer!
- Do not copy cdwacc.dll to another directory. Having more than one copy of cdwacc.dll on a system leads to various kinds of problems. The RSJ CD Writer installation folder is added to the system executable path during installation, so applications linked to cdwacc.dll will run from any directory on a system with RSJ CD Writer installed.
- The RSJ CD Writer Conditions of Use require a license for each machine with RSJ CD Writer installed; in other words, a license is needed for each machine the application linked to cdwacc.dll is running on.
- There is no function to write data to a CD in this API. If you miss it, remember that RSJ CD Writer provides a CD recorder file system. Use the normal file access functions of your programming environment (CreateFile, fopen, Read/Write,...).

RSJ CD Writer API functions

Notes:
Most functions expect a drive letter as input parameter. This should be passed in the format

```
<drive_letter>:
```

for example "`F:`" (not case sensitive).

All functions return an error code of type *CDWACCRET* (defined as "short"). For a list of possible error codes, please see the Return Codes chapter.

The following functions are available:

Function	**Purpose**
cdwGetDrvLetters	Returns an array of recorder drive letters for this system
cdwGetDevCaps	Returns device capabilites of a recorder drive
cdwGetParms	Returns current file system settings
cdwSetParms	Sets file system settings
cdwFormat	Formats a CD medium (obsolete)
cdwUmount	Unmounts (finalizes) a CD
cdwUnseal	Removes write protection from a CD
cdwEraseCDRW	Erases a CD-RW
cdwGetCDInfo	Returns CD information
cdwGetVersion	Returns the CD Writer API version
cdwSetBootfile	Makes a CD bootable (ElTorito)
cdwQueryCacheStatus	Returns the file system cache fill state

cdwGetDrvLetters

Purpose: Returns all drive letters handled by the CD Writer File System (CDWFS).

Syntax: `cdwGetDrvLetters (DRVMAP drvmap);`

Returns: See chapter Return Codes

Remarks:

This function returns all drive letters handled by the CD Writer File System. The DRVMAP type is defined as a character array (see "cdwacc.h" for more details). The characters returned in DRVMAP are all uppercase. For example, if 'DRVMAP' contains "DEF", drives D:, E: and F: are CDWFS drives.

The DRVMAP is terminated with a 0x00 character.

Example code:

```
char           cdw_drives[sizeof(DRVMAP)][5];

CDWACCRET GetCDWriterDriveLetters(char **cdw_drives)
{
DRVMAP         drvmap;
int            i;
CDWACCRET      ret;

ret = cdwGetDrvLetters(drvmap);
if (ret != CDWACC_OK) {
  return(ret);
  }

/* scan the drvmap returned for CDWFS drives */
for (i = 0; drvmap[i] != '\0'; i++) {
  /* build valid drive letter including colon */
  sprintf(cdw_drives[i], "%c:", drvmap[i]);
  }

return(CDWACC_OK);
}
```

cdwGetDevCaps

Purpose: Returns the device capabilities of the CDWFS drive specified

Syntax:

```
cdwGetDevCaps (char *drive_ltr,
                  CDWACC_DEV_CAPS *pdev_caps);
```

Returns: See chapter Return Codes

Remarks:

There does not have to be a CD inserted for this function to succeed. However, if a CD is inserted, the returned results are more accurate since they reflect the medium type (speed table).

The CDWACC_DEV_CAPS structure is defined as follows:

```
typedef struct {
  long        cb;                /* size of structure */
  long        removable;         /* device uses removable media */
  long        writable;          /* device is able to write data */
  long        session;           /* device supports (or needs) sessions */
  long        rmtrack;           /* device can remove single tracks */

  /* audio parameters (not used with the file system) */
  long        read_audio;        /* device can read audio tracks */
  long        play_audio;        /* device can play audio tracks */

  long        speed_count;       /* number of entries in  'speeds_table' */
  long        speed_table[20];   /* array with possible recording speed factors */

  long        dev_type;          /* type of CD recorder */

  } CDWACC_DEV_CAPS;
```

All members are defined as *long* to ensure Visual Basic compatibility.

Member	**Direction**	**Purpose**
cb	input	length of this structure. Must be filled in before calling cdwGetDevCaps()
removable	output	if not zero, the device uses removable media
writable	output	if not zero, the drive is able to write data
session	output	if not zero, the drive supports (needs) sessions
rmtrack	output	if not zero, the drive can remove single tracks
read_audio	-	currently not used
play_audio	-	currently not used
speed_count	output	specifies the number of entries in the *speed_table* array
speed_table	output	receives a list of possible recording speed factors (1 means 150K/sec, 2 = 300K/sec,...)

Member	Direction	Purpose
dev_type	output	specifies the type of the drive. This can be one of the following: `CDWACC_DVT_CDROM` CD-ROM drive `CDWACC_DVT_CDREC` CD recorder `CDWACC_DVT_CDRW` CD-RW recorder `CDWACC_DVT_HD` hard disk The CD-ROM and hard disk types are currently not used with the Windows version of the API.

Example code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


typedef unsigned char BYTE;  /* normally declared in windows.h */
#include "cdwacc.h"


int main(int argc, char **argv)
{
CDWACCRET ret;
CDWACC_DEV_CAPS dev_caps;
DRVMAP drvmap;
char drv[5];

/* get available recorders; DRVMAP will be zero terminated */
cdwGetDrvLetters(drvmap);
if (drvmap[0] == '\0') {
  printf("No recorders found!\n");
  return(1);
  }

/* create drive letter as expected by RSJ CD Writer API */
sprintf(drv, "%c:", drvmap[0]);
memset(&dev_caps, 0x00, sizeof(dev_caps));
dev_caps.cb = sizeof(dev_caps);
ret = cdwGetDevCaps(drv, &dev_caps);

if (ret) {
  printf("cdwGetDevCaps() returned %d\n");
  } else {
  printf("cdwGetDevCaps() succeeded.\n");
  }

return(0);
}
```

cdwGetParms

Purpose: Returns the current operating parameters of the CDWFS drive specified. These parameters match those accessible in the drive’s property page.

Syntax:

```
cdwGetParms (char *drive_ltr,
                 CDWACC_FSPARMS *pfsparms);
```

Returns: See chapter Return Codes

Remarks:

The CDWACC_FSPARMS structure is defined as follows:

```
typedef struct {
  long                cb;         /* size of structure */

  /* parameters which can be modified anytime */
  long                debug;      /* print debug messages */
  long                eject;      /* eject CD after finalizing */
  long                nolock;     /* don't lock drive door while
                                     CD is mounted */
  long                speed;      /* speed factor (1 = 150K, 2 = 300K, ...) */
  long                emulation;  /* emulation write */
  long                nozap;      /* prevent overwriting or deleting files
                                     not implemented yet in Win95 */
  long                tmode;      /* track mode for next track */
  long                iso_level;  /* ISO9660 Information Interchange Level
                                     (determines how ISO9660 filenames are built
                                      when closing the CD) */

  /* parameters which can only be changed during 'fsMount()' */
  unsigned long       cache_1;     /* size of RAM cache (in KB) */
  unsigned long       cache_2;     /* size of second level cache
                                      (ignored on Windows systems) */
  char                cache_path[260];
                                   /* path where the 2nd-level cache
                                       files will be placed
                                       (ignored on Windows systems) */
  /* parameters which can only be read */
  long                ver_major;  /* major file system version number */
  long                ver_minor;  /* minor file system version number */
   } CDWACC_FSPARMS;
```

All members are defined as *long* to ensure Visual Basic compatibility.

Member	**Direction**	**Purpose**
cb	input	length of this structure. Must be filled in before calling cdwGetParms()
debug	output	does not apply to the Windows version of the API
eject	output	if set, the CD is ejected after finalization. Some recorders require this to be set.
nolock	output	if set, the drive door is not locked during sessions. This is dangerous, since ejecting the CD before the cache is flushed ruins the CD
speed	output	current speed factor (1 => 150k/sec)
emulation	output	if set, the recorder does not actually write. This can be used to verify that the data source is fast enough for a certain recording speed.

Member	Direction	Purpose
nozap	output	if set, files on the CD cannot be deleted or overwritten. This is not yet implemented in the Windows version of the API.
tmode	output	specifies the track mode being used. Available modes are: `CDWACC_TM_MODE1` (CD-ROM mode) `CDWACC_TM_MODE2` (CD-XA mode, recommended)
iso_level	output	specifies the ISO level being used. Possible values are 1 - 8.3 filenames 2 - Allows up to 31 characters for file names; each file name must contain a period (.). 3 - Allows 31 characters for file names and supports special characters, e.g. ä-ö-ü-ß and so on.
cache_1	output	specifies the amount of RAM in kB to be used for data caching.
cache_2	output	does not apply to the Windows version
cache_path	output	does not apply to the Windows version
ver_major	output	major file system version number
ver_minor	output	minor file system version number

This structure is also used with the cdwSetParms function (see below). In this case, all members are input and the ver_major and ver_minor members are ignored.

cdwSetParms

Purpose: Sets the operating parameters of the CDWFS drive specified. These parameters match those accessible in the drive's property page.

Syntax:

```
cdwSetParms (char *drive_ltr,
             CDWACC_FSPARMS *pfsparms);
```

Returns: See chapter Return Codes

Remarks:

For a description of the CDWACC_FSPARMS structure, please see cdwGetParms above.

cdwFormat

Purpose: Formats the CD in the drive specified.

Syntax:

```
cdwFormat (char *drive_ltr);
```

Returns: See chapter Return Codes

Remarks:

This function is obsolete. Do not call it anymore in new projects.

cdwUmount

Purpose: Unmounts or closes a CD.

Syntax:

```
cdwUmount (char *drive_ltr,
           CDWACC_FINALIZE_MODE mode,
           CDWACC_VOL_INFO *pvol_info);
```

Returns: See chapter Return Codes

Remarks:

Available finalize modes are:

`CDWACC_FINALIZE_NONE`	emergency eject – no buffers are flushed, no directories written!
`CDWACC_FINALIZE_CACHE`	not needed anymore – do not use this option
`CDWACC_FINALIZE_DIRECTORY`	write cache buffer and directory to disk
`CDWACC_FINALIZE_SESSION`	same as CDWACC_FNIALIZE_DIRECTOY, but additionally writes PVD and session header to the CD
`CDWACC_FINALIZE_SEAL`	same as CDWACC_FINALIZE_SESSION, but additionally seals the CD (write protection)

The CDWACC_VOL_INFO parameter is currently not used and should be NULL. This feature will be added in a future version.

cdwUnseal

Purpose: Removes the write protection of a CD.

Syntax:

```
cdwUnseal (char *drive_ltr);
```

Returns: See chapter Return Codes

Remarks:

This command removes the write protection from a CD that was unmounted with the CDWACC_FINALIZE_SEAL option. After successfully calling this function, files can again be written to the CD.

cdwEraseCDRW

Purpose: Completely erases a CD-RW medium in a CD-RW recorder.

Syntax: `cdwEraseCDRW (char *drive_ltr);`

Returns: See chapter Return Codes

Remarks:

This function completely erases all contents of a CD-RW medium in a CD-RW drive. This command can not be undone.

cdwGetCDInfo

Purpose: Receives information about a CD. This includes track/session information and information about the number of files and directories on the CD.

Syntax:

```
cdwGetCDInfo(char *drive_ltr,
                CDWACC_CD_INFO *pcd_info);
```

Returns: See chapter Return Codes

Remarks:

The CDWACC_CD_INFO structure is defined as follows:

```
typedef struct {
  long    cb;                    /* size of structure */
  long    file_count;            /* number of files on the CD */
  long    dir_count;             /* number of directories on the CD */
  long    file_disk_usage;       /* volume space occupied by files */
  long    dir_disk_usage;        /* volume space occupied by directories */

  long    finalized_sessions;    /* number of finalized sessions on the CD */
  long    open_session;          /* currently open session */
  long    track_count;           /* number of tracks on the CD */
  long    reserved_track;        /* currently reserved track */
  long    fixation_recommended;  /* power calibration area almost full */
  long    modified;              /* CD has been modified */

  } CDWACC_CD_INFO;
```

All members are defined as *long* to ensure Visual Basic compatibility.

Member	Direction	Purpose
cb	input	length of this structure. Must be filled in before calling cdwGetCDInfo()
file_count	output	number of files on the CD
dir_count	output	number of directories on the CD
file_disk_usage	output	number of bytes occupied by files
dir_disk_usage	output	number of bytes occupied by directories

Member	Direction	Purpose
finalized_sessions	output	number of finalized sessions on the CD
open_session	output	index of the open session (if any); 1-based
track_count	output	number of tracks
reserved_track	output	index of reserved track (if any); 1-based
fixation_recommended	output	if not zero, the CD should be finalized as soon as possible
modified	output	if set, the CD has been modified since the last unmount

cdwGetVersion

Purpose: Returns the version of the CD Writer API module (cdwacc.dll)

Syntax:

```
cdwGetVersion(BYTE *major,
              BYTE *minor);
```

Returns: See chapter Return codes

Remarks:

To retrieve the version of the CDWFS file system, use the cdwGetParms function.

cdwSetBootfile

Purpose: Makes a CD bootable (ElTorito)

Syntax:

```
cdwSetBootfile  (char *drve_ltr,
                 char *boot_image_file);
```

Returns: See chapter Return codes

Remarks:

This function marks a file on the CD as the boot image file. The file must be located on the CD before calling this function. The CD has to be finalized for the changes to take effect.

cdwQueryCacheStatus

Purpose: Returns status information about the current file system cache usage.

Syntax:

```
cdwQueryCacheStatus (char *drive_ltr,
                     unsigned long *cache_size,
                     unsigned long *used_cache);
```

Returns: See chapter Return codes

Remarks:

Currently, the unit used for 'cache_size' and 'cache_status' depends on the recorder driver and is subject to change, so this information is useful for percent-like calculations only. See the example code below.

Example:

```
unsigned long GetCacheFilledPerCent(char *drv_ltr)
{
unsigned long cache_size;
unsigned long used_cache;

if (cdwQueryCacheStatus(drv_ltr, &cache_size, &used_cache) != CDDWACCRET_OK) {
  return(0);
  }

return(used_cache * 100 / cache_size);
}
```

Return Codes

Following now is a short description of each of the available return codes. Please see the „cdwacc.h“ file for latest additions.

Note: Although defined in groups, expect each error to be returned by any call.

CDWACC_OK	0	no error
General Errors		
CDWACC_FAILED_TO_OPEN	1	failed to open VXD
CDWACC_INVALID_PARM	2	invalid parameter was passed
Opening The Device		
CDWACC_DEVICE_NOT_FOUND	100	device could not be found
CDWACC_FILE_NOT_FOUND	101	file could not be found
CDWACC_PATH_NOT_FOUND	102	path could not be found
CDWACC_ACCESS_DENIED	103	access denied for some reason
CDWACC_DEVICE_LOCKED	104	device locked by another process
CDWACC_INVALID_DEVICE	105	invalid device type
CDWACC_INVALID_DRIVER	106	driver module in error
CDWACC_NOT_READY	107	device not ready
CDWACC_OUTOFMEM	108	out of memory
Working With The Device		
CDWACC_ARENA_TRASHED	200	internal control structures damaged
CDWACC_INVALID_HANDLE	201	invalid device handle
CDWACC_INVALID_FUNC	202	invalid function
CDWACC_INVALID_DATA	203	invalid data for function
CDWACC_NOT_SUPPORTED	204	function not supported
CDWACC_IOCTL_ERROR	205	generic IOCtl error
CDWACC_UNIT_ATTENTION	206	illegal medium change
CDWACC_RECORDER_BUSY	207	recorder cannot process the requested command at this time
CDWACC_INTERRUPTED	208	the command has been interrupted
CDWACC_NOBLANKIFCHANGED	209	CD has been modified in current session and cannot be blanked
Reading And Writing		
CDWACC_WRITE_PROTECT	300	medium and/or device write protected
CDWACC_DISK_FULL	301	disk full
CDWACC_CRC_ERROR	302	data error (readorwrite)
CDWACC_SEEK_ERROR	303	sector address out of range
CDWACC_READ_ERROR	304	logical read error
CDWACC_WRITE_ERROR	305	logical write error
CDWACC_GEN_FAILURE	306	general failure

CDWACC_FLUSH_FAILED	307	FLUSH CACHE command failed, but drive is not in write mode anymore

Copying Tracks

CDWACC_BUFFER_UNDERRUN	400	data could not be written in time
CDWACC_COPY_ABORTED	401	operator has cancelled the copy operation
CDWACC_BUFFER_OVERFLOW	402	audio data could not be read in time

File System Errors

CDWACC_INVALID_FORMAT	500	invalid file system format
CDWACC_NO_MORE_HANDLES	501	no more handles available (out of memory)
CDWACC_NO_MORE_FILES	502	no more files in this directory
CDWACC_INVALID_DIRECTORY	503	invalid directory in path name
CDWACC_DIR_NOT_EMPTY	505	directory is not empty (rmdir)
CDWACC_NEGATIVE_SEEK	506	negative seek offset
CDWACC_SHARING_VIOLATION	507	file already used by someone else
CDWACC_ADDRESS_CHANGED	508	locked cache file or write handle could not be written at the predetermined address
CDWACC_CACHE_FULL	509	the 2nd-level cache is full
CDWACC_FILE_TOO_SMALL	510	file is too small to be copied directly (use standard copy algorithm instead)