The RSJ CD Writer API

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

Note: The API has been created and tested for the Micrsosoft C6.00 and IBM CSet/2 und CSet++ compilers.

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

CDWFSCTL.H

The header file „cdwfsctl.h" has to be included in each module that uses the CD Writer API. It defines the types and constants that are used to communicate with the file system.

For the latest information about eventual API changes, please take a closer look at this file! The examples in this document might not be as up-to-date as the information in the CDWFSCTL.H file.

The following functions are available:

CDWFS_ATTACH

The *CDWFS_ATTACH* command is used to attach a recorder under the drive letter provided. The *ATTACH_INFO* structure contains the information required by this command:

```
 /********************************************************************************

  'ATTACH_INFO' contains the information which is used to attach a drive
  letter to the file system

********************************************************************************/

typedef struct {
  short       len;              /* length of this structure */
  char device[20];              /* name of the SCSI device */
  short       sessions_to_skip; /* number of sessions to skip (open last...) */
  short       formatted;        /* != 0, if medium is formatted */
  } ATTACH_INFO;
```

Structure members:

len	Input. Length of structure
device	Input. Name of the SCSI device driver. Usually, this is RSJSCSI$.
sessions_to_skip	Input. Number of sessions to skip. 0 = current session, 1 = previous session, ...
formatted	Output. Indicates if the CD is already formatted.

Example:

```
#include <stdlib.h>

#define INCL_BASE
#include <os2.h>

#include <cdwfsctl.h>

main()
{
ATTACH_INFO attach_info;
APIRET ret;
/* initialize attach info */
attach_info.len = sizeof(ATTACH_INFO);
attach_info.sessions_to_skip = 0;

strcpy(attach_info.device, "RSJSCSI$");

/* attach drive Z: */
ret = DosFSAttach("z:",
                  "cdwfs",
                  (void *) &attach_info,
                  sizeof(ATTACH_INFO),
                  CDWFS_ATTACH);

/* check return code */
if (ret == NO_ERROR) {
  printf("success\n");
  } else {
  printf("error code: %d\n", (int) ret);
  }
}
```

CDWFS_DETACH

The *CDWFS_DETACH* command is used to finalize or close a CD. Necessary data is passed within a *DETACH_INFO* structure:

```
/*******************************************************************************

  ’DETACH_INFO’ contains the information which is written into the primary
  volume descriptor of the CD when ’flush_mode’ is greater that FLUSH_CACHE.

*******************************************************************************/

typedef struct {
  short       len;                  /* length of this structure */
  FLUSH_MODE  flush_mode;           /* type of flush requested */
  char        vol_set_id[128];      /* volume set identifier */
  char        publisher_id[128];    /* publisher identifier */
  char        preparer_id[128];     /* data preparer identifier */
  char        app_id[128];          /* application identifier */
  char        cpyrght_file[37];     /* name of copyright file in root */
  char        abstrct_file[37];     /* name of abstract file in root */
  char        biblio_file[37];      /* name of bibliographic file in root */
  } DETACH_INFO;
```

Structure members:

len	Input. Length of structure
flush_mode	Input. Can be one of the following values:

FLUSH_NONE	The drive is detached without flushing any buffers (emergency eject).
FLUSH_CACHE	The drive is detached after buffers have been flushed. Since the directory information is not updated, data on the CD cannot be accessed. This mode is used internally and is normally not needed.
FLUSH_DIRECTORY	Writes buffers to the CD and updates the directory information before detaching. This is the same as the “-c” option for the ‘cdattach’ program.
FLUSH_SESSION	Same as FLUSH_DIRECTORY, but the current session is closed and the next opened. Same as “cdattach <dive> -s”.
FLUSH_SEAL	Same as FLUSH_SESSION, but after opening the new session no track is being reserved. The CD is “write protected”. The write protection can be removed using “format /UNSEAL”.

vol_set_id, publisher_id, preparer_id, app_id, app_id, cpyrght_file, abstrct_file, biblio_file

Input. These fields are stored in the Primary Volume Descriptor; the "chkdsk <drive> /V" command prints this information.

Example:

```
#include <stdlib.h>

#define INCL_BASE
#include <os2.h>

#include <cdwfsctl.h>

main()
{
static DETACH_INFO detach_info;
APIRET ret;

/* initialize detach info */
detach_info.len = sizeof(DETACH_INFIO);
detach_info-flush_mode = FLUSH_SESSION

strcpy(detach_info.vol_set_id, "My first CD");
strcpy(detach_info.publisher_id, "RSJ Software GmbH");
strcpy(detach_info.preparer_id, "Bugs Bunny");
strcpy(detach_info.app_id, "RSJ CD-Writer File System");
strcpy(detach_info.cpyrght_file, "");
strcpy(detach_info.abstrct_file, "");
strcpy(detach_info.biblio_file, "");

/* detach drive Z: */
ret = DosFSAttach("z:",
                  "cdwfs",
                  (void *) &detach_info,
                  sizeof(DETACH_INFO),
                  CDWFS_DETACH);

/* check return code */
if (ret == NO_ERROR) {
  printf("success\n");
  } else {
  printf("error code: %d\n", (int) ret);
  }
}
```

CDWFS_SET_SPEED

CDWFS_SET_SPEED sets the writing speed of the recorder. It uses the *SPEED_INFO* structure:

```
/*******************************************************************************

  'SPEED_INFO' is used to specify the recording speed as well as the write
  mode (emulation write or physical write).

*******************************************************************************/

typedef struct {
  short       speed_factor;         /* 1 = 150K, 2 = 300K, 4 = 600K, ... */
  short       emulation_write;      /* if set, the CD will not be modified */
  } SPEED_INFO;
```

Structure fields:

speed_factor	Input. Specifies the new speed factor (1 = 150KB/s, 2 = 300KB/s, 4 = 600KB/s, ...).
emulation_write	Input. If this is != 0, data is not written to the CD. This feature can be used to verify that data is delivered fast enough for the current recording speed..

Note: In contrast to other FSCtl calls this call can be used in two different ways:

- Specifying a drive letter (i.e.. "z:\\") using FSCTL_PATHNAME. This changes the default speed and the speed of a currently attached recorder (z:\).
- Specifying the file system name (CDWFS) using FSCTL_FSDNAME. This command allows modifying the default recording speed without any recorder being currently attached.

Example:

```

#include <stdlib.h>

#define INCL_BASE
#include <os2.h>

#include <cdwfsctl.h>

main()
{
static SPEED_INFO speed_info;
USHORT parm_len = sizeof(SPEED_INFO);
USHORT data_len = 0;

/* select double speed and no emulation write */
speed_info.speed_factor = 2;
speed_info.emulation_write = 0;

/* call SPEED_INFO entry point in CDWFS */
ret = DosFSCtl(NULL,
               data_len,
               &data_len,
               (PBYTE) &speed_info,
               parm_len,
               &parm_len,
               CDWFS_SET_SPEED,
               "z:\\",
               (HFILE) -1,
               FSCTL_PATHNAME,
               0);

/* check return code */
if (ret == NO_ERROR) {
  printf("success\n");
  } else {
  printf("error code: %d\n", (int) ret);
  }
}
```

CDWFS_FORMAT

The *CDWFS_FORMAT* command formats or “unseals” CDs. It expects a parameter of the FORMAT_MODE type:

FORMAT_EMPTY_MEDIUM	Format only empty CDs.
FORMAT_UNSEAL	Unseal a CD that was detached using the –x switch or the FLUSH_SEAL option.
FORMAT_ERASE_RW_DISK	Erases a RW medium

Example:

```
 #include <stdlib.h>

#define INCL_BASE
#include <os2.h>

#include <cdwfsctl.h>
```

```
main()
{
FORMAT_MODE format_mode = FORMAT_EMPTY_MEDIUM;
USHORT data_len = 0;
USHORT parm_len = sizeof(FORMAT_MODE);

/* call FORMAT entry point in CDWFS */
ret = DosFSCtl(NULL,
               data_len,
               &data_len,
               (PBYTE) &format_mode,
               parm_len,
               &parm_len,
               CDWFS_FORMAT,
               "z:\\",
               (HFILE) -1,
               FSCTL_PATHNAME,
               0);

/* check return code */
if (ret == NO_ERROR) {
  printf("success\n");
  } else {
  printf("error code: %d\n", (int) ret);
  }
}
```

CDWFS_CHKDSK

The *CDWFS_CHKDSK* command retrieves information that is displayed by the ‘chkdsk’ OS/2 command. It uses the *CHKDSK_DATA* structure:

```
/*******************************************************************************

  'CHKDSK_DATA' defines the information which is returned by the
  FSCTL_CHKDSK request.

*******************************************************************************/

typedef struct {
  char        copyright[100];       /* copyright string with version information */
  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 */
  short       finalized_sessions;   /* number of finalized sessions on the CD */
  short       open_session;         /* currently open session */
  short       track_count;          /* number of tracks on the CD */
  short       reserved_track;       /* currently reserved track */
  short       fixation_recommended; /* power calibration area almost full */
  short       modified;             /* CD has been modified */
  DETACH_INFO pvd_info;             /* information about the PVD */
  } CHKDSK_DATA;
```

Structure members:

copyright	Output. Contains a copyright notice
file_count	Output. Number of files on the CD
dir_count	Output. Number of subdirectories
file_disk_usage	Output. Number of bytes occupied by files

dir_disk_usage	Output. Number of bytes occupied by directories
finalized_sessions	Output. Number of closed sessions on the CD
open_session	Output. Number of the current session. If this is 0, the CD is either full or a CDROM. In any case this CD cannot be written to.
track_count	Output. Number of tracks on the CD
reserved_track	Output. Number of the reserved track. If this matches the *track_count* value, no files have been written to the CD after it was closed with “cdattach –s”. If this value is 0, the CD is either full, write protected (FLUSH_SEAL or cdattach –x) or it is a CDROM.
fixation_recommended	Output. If not zero, the CD has been modified so many times that the power calibration area is almost full. Closing the current session is urgently recommended because an exhausted power calibration area prevents writing any data to the CD.
modified	Output. If not zero, the CD was modified since it was attached..
pvd_info	Output. This field contains the user information of the Primary Volume Descriptor which is displayed by the command 'chkdsk <Drive> /v'.

Example:

```
 #include <stdlib.h>

#define INCL_BASE
#include <os2.h>

#include <cdwfsctl.h>

main()
{
static CHKDDSK_DATA chkdsk_data;
USHORT data_len = sizeof(CHKDSK_DATA);
USHORT parm_len = 0;

/* call CHKDSK entry point in CDWFS */
ret = DosFSCtl((PBYTE) &chkdsk_data,
               data_len,
               &data_len,
               NULL,
               parm_len,
               &parm_len,
               CDWFS_CHKDSK,
               "z:\\",
               (HFILE) -1,
               FSCTL_PATHNAME,
               0);

/* check return code */
if (ret == NO_ERROR) {
  printf("success\n");
  } else {
  printf("error code: %d\n", (int) ret);
  }
}
```

CopyToCD()

The CopyToCD() function copies a single file to the CD. This function allows writing files which are larger than the cache size into a single track. Further information about this topic can be found in the description of the *cdcopy* command in the RSJ CD Writer Manual.

The function CopyToCD() is located in the .DLL file cdwcpy.dll. In order to use this function, the import library cdwcpy.lib must be linked..

Syntax:

```
#include <cdwfsctl.h>

extern CDW_LINKAGE CopyToCD (char *source,
                             char *target);
```

Parameters:

source	Input. Fully qualified name of the source file
target	Input. Fully qualified name of the target file

Example:

```
#include <stdio.h>

#include "cdwfsctl.h"

main()
{
APIRET ret;

/* copy a huge file into a single track */
ret = CopyToCD("c:\\data\\largefile.dat", "z:\\largefile.dat");

/* check return code */
if (ret == NO_ERROR) {
  printf("success\n");
  } else {
  printf("error code: %d\n", (int) ret);
  }
}
```

XCopyToCD()

The XCopyToCD() function copies complete directory trees to a CD. This function allows writing files which are larger than the cache size into a single track. Further information about this topic can be found in the description of the *cdcopy* command.

Note: The target directories are created automatically, if required. The function XCopyToCD() is located in the .DLL file cdwcpy.dll. In order to use this function, the import library cdwcpy.lib must be linked..

Syntax:

```
#include <cdwfsctl.h>

extern CDW_LINKAGE XCopyToCD (char *source,
                              char *target);
```

Parameter:

source	Input. Fully qualified name of the source file. Wildcards ("?" oder "*") are supported.
target	Input. Fully qualified name of the target directory. Wildcards and filenames are not supported.

Example:

```
#include <stdio.h>

#include <cdwfsctl.h>

main()
{
APIRET ret;

/* copy complete directory tree to the CD */
ret = XCopyToCD("c:\\os2\\*", "z:\\os2bkup");

/* check return code */
if (ret == NO_ERROR) {
  printf("success\n");
  } else {
  printf("error code: %d\n", (int) ret);
  }
}
```

XCopyToCD2()

This function is the same as *XCopyToCD()*, except for the additional ‘verbose’ parameter. See the XCopyToCD function above for further details.

Syntax:

```
#include <cdwfsctl.h>

extern CDW_LINKAGE XCopyToCD2 (char  *source,
                               char  *target,
                               short  verbose);
```

Parameter:

verbose	Input. If not zero, all filenames are printed to ‘stdout’ before being copied.