Data File Interface Library, Programmer's Manual
CU-QMW-MA-0010
16 April 2002

A.J. Allen
Astronomy Unit, Queen Mary and Westfield College,
Mile end Road, London E1 4NS, U.K.
email: A.J.Allen@qmw.ac.uk

April 16, 2002

Contents

Chapter 1

Introduction

This manual describes the family of C routines used in direct data file access and export from within QSAS and the format exchange software QTRAN.

1.0.1 Development

Developed under Sun Solaris 2.7, using ANSI C. New data file formats are added periodically.

Chapter 2

Qie Module

2.1 Introduction

This chapter deals with the functions associated with writing data files from QSAS, and direct reading and browsing of data files by QSAS. Recognised file types are Common Data Format (CDF) files and ascii flat files using the QMW defined syntax (CU-QMW-TN-0011) and Cluster Exchange Files (CEF).

Data entries in flat files may be either separated be delimiters, or arranged by column position in tabular form. Flat files must have associated headers that may be in separate files (detached) or at the start of the data file itself (attached). Details of this syntax are in the document CU-QMW-TN-0011.

2.2 Structures

```
typedef enum QiSOptions_f_type {
  UNSET,
  TABULAR,
  DELIMITED,
  EXCHANGE
} QiOptions_f_type;

typedef enum QiSOptions_header {
  ATTACHED,
  DETACHED,
  NO_HEADER
} QiOptions_header;

typedef enum QiSOptions_rec_num {
  NUM_OFF,
  NUM_ON
} QiOptions_rec_num;

typedef enum QiSOptions_priority {
  REPLACE=1,
  WARN
} QiOptions_priority;

typedef enum QiSOptions_object {
  TS,
  DS
} QiOptions_object;
```

```
typedef enum QiSTimeFormat_e {   /* cope with more time formats  SJS*/
  NOT_A_TIME,
  ISO,
  FREE_TIME_FORMAT
} QiFTTimeFormat_e;

typedef struct QiSOptions
{
  QiOptions_header header;  // flag ATTACHED, DETACHED or no header
  QiOptions_f_type f_type;  // if UNSET uses file clues
  QiOptions_rec_num rec_numbering; // flag to set record numbering on
  QiOptions_priority priority;     // flag for overwrite on new file
  QiOptions_object object_type;         // qsas obj type
  char attr_delim;          // delimiter to use in header
  char data_delim;          // delimiter to use in data
  char rec_end;             /* end of record delimiter */
  char row_end;             /* end of row in array delimiter */
  QiOptions_f_type type_guess;  // file type guess from context
  FILE * fp_display;        // file pointer for output display
  FILE * fp_null;           // file pointer to null for losing things
  char debug_choice;        // 'f'=fp_display, 'w'=QSAS fn, else off
  double sample_H_interval; // sampling interval, normally Half_interval
  char * start_after;       // str to identify last line before data
  char time_sep_attr;       // char separator in ISO time in attrs
  char time_sep_data;       // char separator in ISO time in data recs
  char * header_path;       // path to header file if detached
  char * header_name;       // name of header file if detached
  char * get_only;          // name of single variable to fetch,
                            //    if STR_NULL gets all

  char EXTN_CDF[5];         // strings for known file extensions
  char EXTN_QFT[5];         // these are initialised in QiMakeOptionsObj
  char EXTN_QFD[5];
  char EXTN_QHD[5];
  char EXTN_CEF[5];

} QiOptions;

typedef struct QiSCDFepoch{                /* moved by SJS re FT */
                double tsince0;
                long   year;
                long   month;
                long   day;
                long   hour;
                long   minute;
                long   second;
                long   msec;
} cdf_epoch;

typedef struct QiSFTParser              /* info to parse FT string */
{
   int             n_found_date;        /* number of date strings to get */
   QiFTdatestrings_e  found_date[N_FT_DATE_STRINGS];   /* which ones found  */
   int             date_start_in_FT[N_FT_DATE_STRINGS]; /* start pos (fr 0)*/
   int             date_width_in_FT[N_FT_DATE_STRINGS];  /* field widths */
   int             n_found_time;        /* and all same stuff for time   */
   QiFTtimestrings_e  found_time[N_FT_TIME_STRINGS];
   int             time_start_in_FT[N_FT_TIME_STRINGS];
   int             time_width_in_FT[N_FT_TIME_STRINGS];
} QiFTParser;
```

```
typedef struct QiSFTpacket            /* holds all info; goes into rec fmt */
{
QiFTTimeFormat_e timeformat;
QiFTAllRecordsFlag_e AllRecordsFlag;
char AllRecordsFormatString[MAX_FT_LENGTH];
char AllRecordsTimeString[MAX_FT_LENGTH];
char FreeTimeFormatString[MAX_FT_LENGTH];
cdf_epoch epoch;
QiFTParser ft_parser_s;
double time2msecs_factors_adjusted[N_FT_TIME_STRINGS];
} QiFTpacket;

typedef struct QiSCDFContents
{
  long n_vars;            // number of variables
  long n_recs;            // number of data records
  long time_var_num;      // var number of time variable
  char *io_f_name;        // file name for new file
  char *io_f_path;        // path to directory to hold new file
  char *io_f_extn;        // 4 char file ext, .cdf, .qft, .qfd, .qfb
  long num_g_attrs;       // number of global attrs

  QiCDFVariable ** vardata; // ptr to array of QiSCDFVariable structs
  QiGlobalAttr ** g_attr;   // structure for global metadata

} QiCDFContents;

typedef enum QiSCDFVariable_novary{
  WRITE_ONCE,
  EVERY_RECORD
} QiCDFVariable_novary;

typedef struct QiSCDFVariable
{
  QiCDFVariable_novary novary_opt; // flag to identify Non-RV vars
  long sizeofentry;   // number of bytes for each data entry
  long number;        // variable number in cdf
  char *name;         // variable name
  long data_type;     // CDF data type
  long rec_vary;      // CDF record vary value = VARY or NOVARY
  long max_rec_num;   // record number of last record, start at 0
  long num_dims;      // number of CDF dimensions
  long *dim_sizes;    // ptr to array of sizes for dimensions
  long *dim_varys;    // ptr to array of dim vary's
  char **dim_depends; // ptr to array of Depend_i strs, each dim
  long num_v_attrs;   // number of variable attributes held
  long num_elems;     // number of elements per entry

  void * data;        // pointer to array of data entries

  struct QiSVarAttribute *attribute; // ptr to attr struct

} QiCDFVariable;

typedef struct QiSGAttrEntry
{
  long exists;
  long data_type;
  long num_elems;
  void *data;
} QiGAttrEntry;
```

```
typedef struct QiSGlobalAttr
{
  long number;
  char *name;
  long num_entries;
  struct QcSGAttrEntry *entry;
} QiGlobalAttr;

 typedef struct QiSVarAttribute
  {
    long number;
    char *name;
    long data_type;
    long num_elems;
    void *data;
  } QiVarAttribute;
```

2.3 Routines

2.3.1 QiWriteCSDSgenCDF

QiWriteCSDSgenCDF

Export QiSCDFContents object as CSDS standard CDF file.

Synopsis

```
#include "qie.h"

long QiWriteCSDSgenCDF ( struct QiSCDFContents  * QiSCDF,
                         struct QiSOptions * QiOpt);
```

Parameters

QiSCDF Specifies a pointer to a structure containing the path and name of file to be created as well as the global attributes (metadata) and data together with the corresponding variable attributes. The structure is of type QiSCDFContents.

QiOpt Specifies a pointer to a structure containing processing options. The structure is of type QiSOptions.

Return Value

QMW_OK The file was created and populated successfully.

Otherwise Return error codes are defined in qie.h. See QiErrStr.

Discussion **WriteCSDSgenCDF** creates a CSDS standard CDF file and populates it with the data contained in the structure pointed to by **QiSCDF**. The file name is given by **QiSCDF->io_f_name** and the path by **QiSCDF->io_f_path**.

File processing options are controlled through the structure pointed to by **QiOpt**.

The allocation and freeing of **QiSCDF** and **QiOpt** are the responsibility of the calling routine, and this must be done by calling the associated creation and deletion functions **QiMakeCDFContentsObj**, **QiMakeOptionsObj**, **QiFreeCDFContentsObj** and **QiFreeOptionsObj**. Internal storage is freed before the function returns. The input structures are unaltered.

Related Information

/cluster/devel/include/qie.h

/cluster/devel/lib/qie.o

2.3.2 QiWriteCSDSgenFlat

QiWriteCSDSgenFlat

Write data and metadata from QiSCDFContents object into a flat file.

Synopsis

```
#include "qie.h"

long QiWriteCSDSgenFlat ( struct QiSCDFContents  * QiSCDF,
                          struct QiSOptions * QiOpt);
```

Parameters

QiSCDF Specifies a pointer to a structure containing the path and name of file to be created as well as the global attributes (metadata) and data together with the corresponding variable attributes. The structure is of type QiSCDFContents.

QiOpt Specifies a pointer to a structure containing processing options. The structure is of type QiSOptions.

Return Value

QMW_OK The file was created and populated successfully.

Otherwise Return error codes are defined in qie.h. See QiErrStr.

Discussion **QiWriteCSDSgenFlat** creates a QMW standard ascii file and populates it with the data contained in the structure pointed to by **QiSCDF**. The file name is given by **QiSCDF->io_f_name** and the path by **QiSCDF->io_f_path**.

File processing options are controlled through the structure pointed to by **QiOpt**.

The allocation and freeing of **QiSCDF** and **QiOpt** are the responsibility of the calling routine, and this must be done by calling the associated creation and deletion functions **QiMakeCDFContentsObj**, **QiMakeOptionsObj**, **QiFreeCDFContentsObj** and **QiFreeOptionsObj**. Internal storage is freed before the function returns. Unused pointers within the structures should *not* be set to NULL as these and defaults options are handled safely within the "Make" and "Free" functions. The input structures are unaltered.

Related Information

/cluster/devel/include/qie.h

/cluster/devel/lib/qie.o

2.3.3 QiGetCSDSgenCDF

QiGetCSDSgenCDF

Import CSDS standard CDF file into **QiSCDFContents** object.

Synopsis

```
#include "qie.h"

int QiGetCSDSgenCDF ( const char * filename,
                      struct QiSCDFContents  * QiSCDF);
```

Parameters

filename Specifies the file path and name of the CDF file to be read.

QiSCDF Specifies a pointer to a structure to hold the imported global attributes (metadata) and data together with the corresponding variable attributes. The structure is of type QiSCDFContents.

Return Value

QMW_OK The file was created and populated successfully.

Otherwise Return error codes are defined in qie.h. See QiErrStr.

Discussion **QiWriteCSDSgenFlat** reads a CSDS standard CDF file and populates the structure pointed to by **QiSCDF**.

The allocation and freeing of **QiSCDF** is the responsibility of the calling routine, and this must be done by calling the associated creation and deletion functions **QiMakeCDFContentsObj** and **QiFreeCDFContentsObj**. Allocation and freeing of the string **filename** is the responsibility of the calling module. Internal storage is freed before the function returns.

Related Information

/cluster/devel/include/qie.h

/cluster/devel/lib/qie.o

2.3.4 QiGetCSDSgenFlat

QiGetCSDSgenFlat

Read data and metadata from a flat file into QiSCDFContents object.

Synopsis

```
#include "qie.h"

long QiGetCSDSgenFlat ( struct QiSCDFContents  * QiSCDF,
                        struct QiSOptions * QiOpt);
```

Parameters

QiSCDF Specifies a pointer to a structure containing the path and name of file to be read. This structure, on return, will hold the global attributes (metadata) and data together with the corresponding variable attributes. The structure is of type QiSCDFContents.

QiOpt Specifies a pointer to a structure containing processing options. The structure is of type QiSOptions.

Return Value

QMW_OK The file was read and structure populated successfully.

Otherwise Return error codes are defined in qie.h. See QiErrStr.

Discussion **QiGetCSDSgenFlat** reads a QMW standard ascii file and populates the structure pointed to by **QiSCDF**. The file name is given by **QiSCDF->io_f_name** and the path by **QiSCDF->io_f_path**.

File processing options are controlled through the structure pointed to by **QiOpt**.

The allocation and freeing of **QiSCDF** and **QiOpt** are the responsibility of the calling routine, and this must be done by calling the associated creation and deletion functions **QiMakeCDFContentsObj**, **QiMakeOptionsObj**, **QiFreeCDFContentsObj** and **QiFreeOptionsObj**. Internal storage is freed before the function returns. Unused pointers within the structures should *not* be set to NULL as these and defaults options are handled safely within the "Make" and "Free" functions. The input structures are unaltered.

Related Information

/cluster/devel/include/qie.h

/cluster/devel/lib/qie.o

2.3.5 QiFreeCDFContentsObj

QiFreeCDFContentsObj

Free memory allocated by CDF and Flat file read routines for contents structures.

Synopsis

```
#include "qie.h"

long QiFreeCDFContentsObj ( struct QiSCDFContents * QiSCDF);
```

Parameters

QiSCDF Specifies a pointer to a structure to be freed of type QiSCDFContents.

Return Value

QMW_OK Always returned.

Discussion **QiFreeCDFContentsObj** Frees memory allocated by **QiMakeCDFContentsObj()**, and recursively by **QiMakeQiVariablePtrs()**, **QiMakeQiGAttrPtrs()**, **QiMakeQiVariable()**, **QiMakeQiVAttr()**, **QiMakeCharPtrs()** and **QiMakeQiGAttr()** in the structure **QiSCDFContents**, and nested structures. Calls to these related make and free functions are safe against free on unallocated pointers provided that the structure was created using **QiMakeCDFContentsObj**.

Strings should have been created using malloc (or the Qie function **QiNewStr**). If strings are to be freed before calling **QiFreeCDFContentsObj** the associated pointer must be set to the Qie global **STR_NULL** or another malloced string. **QiFreeCDFContentsObj** tests all pointers for NULL before freeing and all strings against STR_NULL using **QistrNULL**.

Related Information

/cluster/devel/include/qie.h

/cluster/devel/lib/qie.o

See also:

QiMakeCDFContentsObj()

QiNewStr()

2.3.6 QiFreeOptionsObj

QiFreeOptionsObj

Free memory allocated for QiSOptions structures.

Synopsis

```
#include "qie.h"

long QiFreeOptionsObj(struct QiSOptions * QiSOpt);
```

Parameters

QiSOpt Specifies a pointer to a structure to be freed of type QiSOptions.

Return Value

QMW_OK Always returned.

Discussion **QiFreeOptionsObj** Frees memory allocated by **QiMakeOptionsObj()** in the structure **QiSOptions**. Calls to these related make and free functions are safe against free on unallocated pointers provided that the structure was created using **QiMakeOptionsObj**.

Strings should have been created using malloc (or the Qie function **QiNewStr**). If strings are to be freed before calling **QiFreeOptionsObj** the associated pointer must be set to the Qie global **STR_NULL** or another malloced string. **QiFreeOptionsObj** tests all pointers for NULL before freeing and all strings against STR_NULL using **QistrNULL**.

Related Information

/cluster/devel/include/qie.h

/cluster/devel/lib/qie.o

See also:

QiMakeOptionsObj()

QiNewStr()

2.3.7 QiMakeCDFContentsObj

QiMakeCDFContentsObj

Malloc memory for QiSCDFContents structures.

Synopsis

```
#include "qie.h"

struct QiSCDFContents * QiMakeCDFContentsObj ( );
```

Parameters

none

Return Value

pointer to space for QiSCDFContents object

Discussion **QiMakeCDFContentsObj** mallocs memory for the structure **QiSCDFContents**. Sub-structures are created using similar functions by the functions called by the "Get" functions. Pointers are initialised to NULL and string pointers to STR_NULL. Calls to these related make and free functions are safe against free on unallocated pointers provided that the structure is freed using **QiFreeCDFContentsObj**.

Strings within the structure should be created using malloc (or the Qie function **QiNewStr**). If strings are to be freed before calling **QiFreeCDFContentsObj** the associated pointer must be set to the Qie global **STR_NULL** or another malloced string. **QiFreeCDFContentsObj** tests all pointers for NULL and all strings against STR_NULL using **QistrNULL** before freeing.

Related Information

/cluster/devel/include/qie.h

/cluster/devel/lib/qie.o

See also:

QiFreeCDFContentsObj()

QiNewStr()

2.3.8 QiMakeOptionsObj

QiMakeOptionsObj

Malloc memory for QiSOptions structures.

Synopsis

```
#include "qie.h"

struct QiSOptions * QiMakeOptionsObj();
```

Parameters

none

Return Value

pointer to space for QiSOptions object

Discussion **QiMakeOptionsObj** mallocs memory for the structure **QiSOptions**. Pointers are initialised to NULL and string pointers to the global constant null string STR_NULL. Calls to these related make and free functions are safe against free on unallocated pointers provided that the structure is freed using **QiFreeOptionsObj**.

Strings within the structure should be created using malloc (or the Qie function **QiNewStr)**. If strings are to be freed before calling **QiFreeOptionsObj** the associated pointer must be set to the Qie global **STR_NULL** or another malloced string. **QiFreeOptionsObj** tests all pointers for NULL and all strings against STR_NULL using **QistrNULL** before freeing.

Related Information

/cluster/devel/include/qie.h

/cluster/devel/lib/qie.o

See also:

QiFreeOptionsObj()

QiNewStr()

2.3.9 QiNewStr

QiNewStr

Create a pointer to space containing string.

Synopsis

```
#include "qie.h"

char * QiNewStr(char * old_str);
```

Parameters

old_str pointer to a string to be copied into the new space.

Return Value

pointer to new string

Discussion **QiNewStr** mallocs memory for the new string and copies old_str into it. This function then returns a pointer to this new string. If malloc fails then a pointer to STR_NULL is returned.

Related Information

/cluster/devel/include/qie.h

/cluster/devel/lib/qie.o

2.3.10 QistrNULL

QistrNULL

Tests string against STR_NULL.

Synopsis

```
#include "qie.h"

int   QistrNULL(char *string);
```

Parameters

string pointer to a string to be tested against the global null string STR_NULL.

Return Value

1 'True' (1) is returned if the string is NULL.

0 'False' (0) is returned if the string is not empty.

Discussion **QistrNULL** uses strcmp() to test string against a NULL string

""

. Empty strings are used (STR_NULL) for unset strings in preference to NULL pointers as they may safely be dereferenced by gui menus etc. This function is then used in place of the test == NULL. Hence QistrNULL(string) is true when string is empty and !QistrNULL(string) is true when string is non-null.

Related Information

/cluster/devel/include/qie.h

/cluster/devel/lib/qie.o

2.3.11 QiErrStr

QiErrStr

Get string associated with error condition.

Synopsis

```
#include "qie.h"

char * QiErrStr (int err\_n);
```

Parameters

err_n the error number returned by another QIE function.

Return Value A string containing an error message relevant to the input error code is always returned.

Discussion **QiErrStr** generates a string for display to the user on encountering one of the QIE eror codes. If the error is associated with a code failure rather than a local problem (such as file write/read permission) then the number is given with "contact CSC support". If the error code is err_n = QMW_OK then the string "OK" is returned. Space for the return string is malloced by QiErrStr and must be freed by the calling function after use.

Related Information

/cluster/devel/include/qie.h

/cluster/devel/lib/qie.o