ADVENTURE_IO

Input/Output format and libraries for ADVENTURE modules

List of Input/Output Functions

February 17, 2006

ADVENTURE Project

Contents

1. Open/Close of *Adv* file

◆ **`AdvDocFile* adv_dio_file_open(const char* filename, const char* mode)`**

Opens the *Adv* file.

returned value		The pointer of **`AdvDocFile`**
Argument	**`filename:`**	The name of the file stored in **`AdvDocFile`**

	`mode:`	The mode showing the purpose to open the file:
		`r` for read (default);
		`c` for create;
		`a` for append.

◆ **`void adv_dio_file_close(AdvDocFile* dfile)`**

Closes the *Adv* file **`dfile`**.

returned value		None
argument	**`dfile:`**	The pointer of **`AdvDocFile`** for closing

◆ **`const char* adv_dio_file_get_locator(AdvDocFile* dfile)`**

Returns the absolute path to the *Adv* file pointed out by **`dfile`**.

returned value		An absolute path
argument	**`dfile:`**	The pointer of **`AdvDocFile`**

2. Open/Close of `AdvDocument`

◆ **`AdvDocument* adv_dio_create(AdvDocFile* dfile, const char* did)`**

Opens a new *Document* in **`dfile`**. *Document ID* **`did`** is created using **`adv_dio_get_documentid`** (see page 6).

returned value		The pointer of **`AdvDocument`** for opening
argument	**`dfile`**:	The pointer of **`AdvDocFile`**
	`did`:	*Document ID* attached to **`AdvDocument`**

◆ **`AdvDocument* adv_dio_open_by_documentid(AdvDocFile* dfile, const char* did)`**

Opens *Document* with specified *Document ID* contained in **`dfile`**.

returned value		The pointer of corresponding **`AdvDocument`**
argument	**`dfile`**:	The pointer of **`AdvDocFile`** (search basis)
	`did`:	*Document ID*

◆ **`AdvDocument* adv_dio_open_nth(AdvDocFile* dfile, int n)`**

Opens the **`n`**th *Document* contained in **`dfile`**.

returned value		The pointer of corresponding **`AdvDocument`**
argument	**`dfile`**:	The pointer of **`AdvDocFile`** (search basis)
	`n`:	An integer

◆ **`AdvDocument* adv_dio_open_by_property(AdvDocFile* dfile, void* prev, ..., NULL)`**

Opens *Document* with specified *Property* contained in **`dfile`**. The keys and values of *Property* should be put in **`“...”`** one-by-one for search operations. If **`prev`** is **`NULL`**, the first matched *Document* will be returned. If the previously matched *Document* is set as a pointer **`prev`**, the next *Document*, which matches with search conditions can be found. **`NULL`** should be added to the end of the search specification.

returned value		The pointer of corresponded **`AdvDocument`**
argument	**`dfile:`**	**`AdvDocFile`** for search
	`prev:`	*Document* matched with conditions in **`“...”`**

	`...` :	Search conditions

Example:

```
doc = adv_dio_open_by_property
( dfile, NULL, "content_type", "FEGenericAttribute",
              "label", "Load", NULL);
```

Document from **dfile** which has *Property* "**content_type=FEGenericAddribute**", "**label=Load**" will be opened.

◆ **`AdvDocument* adv_dio_open_by_locator(const char* locator)`**
Opens *Document* with **`locator`** (a combination of *Document ID* and the path to the file containing *Document* (See page 6)).

returned value		The pointer of corresponding *Document*
argument	**`locator`**:	A sting of characters containing the path to the file and the *Document ID* connected by “?”

◆ **`void adv_dio_close(AdvDocument* doc)`**
Closes *Document*.

returned value		None
argument	**`doc`**:	The pointer of closing *Document*

3. Status of `AdvDocument`

◆ **`const char* adv_dio_make_documentid(const char* str)`**
Creates *Document ID* on the **`str`** basis.

returned value		Created *Document ID*
argument	**`str:`**	The character string for the basis of *Document ID* (For example: **`label@content_type`**)

◆ **`const char* adv_dio_get_documentid(AdvDocument* doc)`**
Returns *Document ID* of *Document* indicated by **`doc`**.

returned value		*Document ID* (a string of characters)
argument	**`doc:`**	The pointer of *Document*

◆ **`adv_off_t adv_dio_get_size(AdvDocument* doc)`**
Returns the size of the *Raw Data* of *Document* indicated by **`doc`**.

returned value		The size of *Document* (an integer)
argument	**`doc:`**	The pointer of *Document*

◆ **`const char* adv_dio_get_locator(AdvDocument* doc)`**
Acquires **`locator`** of *Document* **`doc`**. **`locator`** is a unique string of characters assigned to indicate *Document* by an absolute path to the file and *Document ID* **`doc`** in the form of *Path*?*Document ID*.

returned value		**`locator`** (a string of characters)
argument	**`doc:`**	**`AdvDocument`**

<u>Example:</u>

```
/*********************************
example.c
*********************************/
AdvDatabox *dbox;
AdvDocument *docin;
dbox = adv_dbox_new();
adv_dbox_add(dbox, argv[1]);

docin = adv_dbox_find_by_property(
       dbox,NULL, "label","Displacement",NULL);
/*  Document with Property "label=Displacement" is put into docin */
fprintf(stdout,"%s¥n",adv_dio_get_locator(docin));
```

Output Data:

The path to *Disp.adv* and *Document ID* **docin** are shown.

```
% example Disp.adv
/home/Disp.adv?6B8B4567:Displacement@HDDM_
FEGA:1988:39E5AB59
```

4. Reading and Writing of Properties

◆ **`void adv_dio_set_property(AdvDocument* doc, const char* key, const char* val)`**
Sets the character-type values of *Property* to *Document* **`doc`**.

returned value		None
argument	**`doc:`**	**`AdvDocument`** for set
	`key:`	The item of *Property*
	`val:`	Character-type data which will be set as values

◆ **`void adv_dio_set_property_int32(AdvDocument* doc, const char* key, int32 val)`**
Sets the **`int32`**-type values of *Property* to *Document* **`doc`**.

returned value		None
argument	**`doc:`**	**`AdvDocument`** for set
	`key:`	The item of *Property*
	`val:`	**`int32`**-type data which will be set as values

◆ **`void adv_dio_set_property_float64(AdvDocument* doc, const char* key, float64 val)`**
Sets the **`float64`**-type values of *Property* to *Document* **`doc`**.

returned value		None
argument	**`doc:`**	**`AdvDocument`** for set
	`key:`	The item of *Property*
	`val:`	**`float64`**-type data which will be set as values

◆ **`const char* adv_dio_get_property(AdvDocument* doc, const char* key)`**
Reads the character-type value of *Property* from *Document* **`doc`**.

returned value		Value of *Property* corresponding to **`key`**
argument	**`doc:`**	**`AdvDocument`** for read
	`key:`	The item of *Property*

◆ **`bool adv_dio_get_property_int32(AdvDocument* doc, const char* key, int32* val)`**

Reads the **`int32`**-type value of *Property* from *Document* **`doc`**.

returned value		Normal operation: a value other than **`0`** Error: **`0`** value
argument	**`doc:`**	**`AdvDocument`** for read
	`key:`	The item of *Property*
	`val:`	Pointer of data substitution location (**`int32`**-type)

◆ **`bool adv_dio_get_property_float64(AdvDocument* doc, const char* key, float64* val)`**

Reads the **`float64`**-type value of *Property* from *Document* **`doc`**.

returned value		Normal operation: a value other than **`0`** Error: **`0`** value
argument	**`doc:`**	**`AdvDocument`** for read
	`key:`	The item of *Property*
	`val:`	Pointer of data substitution location (**`float64`**-type)

◆ **bool adv_dio_get_nth_property(AdvDocument* doc, int n, char* key, int keysize, char* val, int valsize)**

Inputs the value of **`n`**th *Document*'s *Property* to **`key`** and **`val`** respectively. **`keysize`** and **`valsize`** are the maximum number of characters that can be assigned for **`key`** and **`val`** (the reserved size of memory array must be large enough to store the data).

returned value		Normal operation: a value other than **`0`** Error: **`0`** value
argument	**`doc:`**	**`AdvDocument`** for read
	`key:`	The pointer indicating the item's name of *Property*
	`keysize:`	Maximum number of characters, which can be assigned for **`key`**
	`val:`	The pointer of variables
	`valsize:`	Maximum number of characters, which can be assigned for **`val`**

Example:

```
AdvDocument *doc
int n = 0;
char key[1024];
char val[1024];
while (adv_dio_get_nth_property
          (doc, n, key, sizeof(key), val, sizeof(val))) n++;
      /*Reads all  Property  */
```

◆ **`void adv_dio_unset_nth_property(AdvDocument* doc, int n)`**

Deletes **`n`** th *Property* of *Document* contained in **`doc`** from memory.

returned value		None
argument	**`doc:`**	The pointer of **`AdvDocument`** for *Property* deletion
	`n:`	An integer

5. Reading of *Raw Data*

◆ **`int32 adv_dio_read_octet(AdvDocument* doc, adv_off_t offset, int32 len, octet* buf)`**

Reads the **`len`** number of the **`octet`**-type (8-bits) data parts from the position specified by **`offset`** in *Raw Data* of *Document* **`doc`**.

returned value		Size of read data
argument	**`doc:`**	*Document* for reading
	`offset:`	The reading position in *Raw Data*
	`len:`	The number of parts of **`octet`**–type data for reading
	`buf:`	The address of stored read **`octet`**-type data

◆ **`int32 adv_dio_read_string_length(AdvDocument* doc, adv_off_t offset)`**

Counts the number of characters in the string at the **`offset`** reading position in *Raw Data* of *Document* **`doc`**.

returned value		Size of read data
argument	**`doc:`**	*Document* for reading
	`offset:`	The reading position in *Raw Data*

◆ **`int32 adv_dio_read_string(AdvDocument* doc, adv_off_t offset, char* buf)`**

Reads the character string data from the position specified by **`offset`** in *Raw Data* of *Document* **`doc`** and stores it in **`buf`**. The size of **`buf`** must be larger than the size of the character string of the data read.

returned value		Size of read data
argument	**`doc:`**	*Document* for reading
	`offset:`	The reading position in *Raw Data*
	`buf:`	The address of stored read data

◆ **`int32 adv_dio_read_int8(AdvDocument* doc, adv_off_t offset, int8* val)`**

Reads the data as 8-bits **`int`**-type from the position specified by **`offset`** in *Raw Data* of *Document* **`doc`**.

returned value		Size of read data
argument	**`doc:`**	*Document* for reading
	`offset:`	The reading position in *Raw Data*
	`val:`	The address of stored read data

◆ **`int32 adv_dio_read_int8v(AdvDocument* doc, adv_off_t offset, int num, int8* val)`**

Reads the **`num`** parts of data as 8-bits **`int`**-type from the position specified by **`offset`** in *Raw Data* of *Document* **`doc`**.

returned value		Size of read data
argument	**`doc:`**	*Document* for reading
	`offset:`	The reading position in *Raw Data*
	`num:`	The number of data parts to be read
	`val:`	The address of stored read data

◆ **`int32 adv_dio_read_int16(AdvDocument* doc, adv_off_t offset, int16* val)`**

Reads the data as 16-bits **`int`**-type from the position specified by **`offset`** in *Raw Data* of *Document* **`doc`**.

returned value		Size of read data
argument	**`doc:`**	*Document* for reading
	`offset:`	The reading position in *Raw Data*
	`val:`	The address of stored read data

◆ **`int32 adv_dio_read_int16v(AdvDocument* doc, adv_off_t offset, int num, int16* val)`**

Reads the **`num`** parts of data as 16-bits **`int`**-type from the position specified by **`offset`** in *Raw Data* of *Document* **`doc`**.

returned value		Size of read data
argument	**`doc:`**	*Document* for reading
	`offset:`	The reading position in *Raw Data*
	`num:`	The number of data parts to be read
	`val:`	The address of stored read data

◆ **`int32 adv_dio_read_int32(AdvDocument* doc, adv_off_t offset, int32* val)`**

Reads the data as 32-bits **`int`**-type from the position specified by **`offset`** in *Raw Data* of *Document* **`doc`**.

returned value		Size of read data
argument	**`doc:`**	*Document* for reading
	`offset:`	The reading position in *Raw Data*
	`val:`	The address of stored read data

◆ **`int32 adv_dio_read_int32v(AdvDocument* doc, adv_off_t offset, int num, int32* val)`**

Reads the **`num`** parts of data as 32-bits **`int`**-type from the position specified by **`offset`** in *Raw Data* of *Document* **`doc`**.

returned value		Size of read data
argument	**`doc:`**	*Document* for reading
	`offset:`	The reading position in *Raw Data*
	`num:`	The number of data parts to be read
	`val:`	The address of stored read data

◆ **`int32 adv_dio_read_int64(AdvDocument* doc, adv_off_t offset, int64* val)`**

Reads the data as 64-bits **`int`**-type from the position specified by **`offset`** in *Raw Data* of *Document* **`doc`**. In the computer environments where 64-bits **`int`**-type is not supported, **`int32*`** will be used for the data type of **`val`**. In this case, only the 32-bits part will be returned.

returned value		Size of read data
argument	**`doc:`**	*Document* for reading
	`offset:`	The reading position in *Raw Data*
	`val:`	The address of stored read data

◆ **`int32 adv_dio_read_int64v(AdvDocument* doc, adv_off_t offset, int num, int64* val)`**

Reads the **`num`** parts of data as 32-bits **`int`**-type from the position specified by **`offset`** in *Raw Data* of *Document* **`doc`**. In the computer environments where 64-bits **`int`**-type is not supported, **`int32*`** will be used for the data type of **`val`**. In this case, only the 32-bits part will be returned.

returned value		Size of read data
argument	**`doc:`**	*Document* for reading
	`offset:`	The reading position in *Raw Data*
	`num:`	The number of data parts to be read
	`val:`	The address of stored read data

◆ **`int32 adv_dio_read_float32(AdvDocument* doc, adv_off_t offset, float32* val)`**

Reads the data as 32-bits **`float`**-type from the position specified by **`offset`** in *Raw Data* of *Document* **`doc`**.

returned value		Size of read data
argument	**`doc:`**	*Document* for reading
	`offset:`	The reading position in *Raw Data*
	`val:`	The address of stored read data

◆ **`int32 adv_dio_read_float32v(AdvDocument* doc, adv_off_t offset, int num, float32* val)`**

Reads the **`num`** parts of data as 32-bits **`float`**-type from the position specified by **`offset`** in *Raw Data* of *Document* **`doc`**.

returned value		Size of read data
argument	**`doc:`**	*Document* for reading
	`offset:`	The reading position in *Raw Data*
	`num:`	The number of data parts to be read
	`val:`	The address of stored read data

◆ **`int32 adv_dio_read_float64(AdvDocument* doc, adv_off_t offset, float64* val)`**

Reads the data as 64-bits **`float`**-type from the position specified by **`offset`** in *Raw Data* of *Document* **`doc`**.

returned value		Size of read data
argument	**`doc:`**	*Document* for reading
	`offset:`	The reading position in *Raw Data*
	`val:`	The address of stored read data

◆ **`int32 adv_dio_read_float64v(AdvDocument* doc, adv_off_t offset, int num, float64* val)`**

Reads the **`num`** parts of data as 64-bits **`float`**-type from the position specified by **`offset`** in *Raw Data* of *Document* **`doc`**.

returned value		Size of read data
argument	**`doc:`**	*Document* for reading
	`offset:`	The reading position in *Raw Data*
	`num:`	The number of data parts to be read
	`val:`	The address of stored read data

6. *Writing of Raw Data*

◆ **`int32 adv_dio_write_octet(AdvDocument* doc, adv_off_t offset, int32 length, const octet* buf)`**
Writes the **`octet`**-type data from **`buf`** into the position specified by **`offset`** in *Document* **`doc`**.

returned value		Size of written data
argument	**`doc:`**	*Document* for writing in
	`offset:`	The writing position in *Raw Data*
	`len:`	The length of **`octet`**-type data to be written
	`buf:`	The **`octet`**-type data

◆ **`int32 adv_dio_write_string(AdvDocument* doc, adv_off_t offset, const char* buf)`**
Writes the string type data from **`buf`** into the position specified by **`offset`** in *Document* **`doc`**.

returned value		Size of written data
argument	**`doc:`**	*Document* for writing
	`offset:`	The writing position in *Raw Data*
	`buf:`	The string data

◆ **`int32 adv_dio_write_int8(AdvDocument* doc, adv_off_t offset, int8 val)`**
Writes the **`int`**-type 8-bits data into the position specified by **`offset`** in *Document* **`doc`**.

returned value		Size of written data
argument	**`doc:`**	*Document* for writing
	`offset:`	The writing position in *Raw Data*
	`val:`	The written data

◆ **`int32 adv_dio_write_int8v(AdvDocument* doc, adv_off_t offset, int num, const int8* val)`**

Writes the **`num`** parts of 8-bits **`int`**-type data into the position specified by **`offset`** in *Document* **`doc`**.

returned value		Size of written data
argument	**`doc:`**	*Document* for writing
	`offset:`	The writing position in *Raw Data*
	`num:`	The number of data parts
	`val:`	The written data

◆ **`int32 adv_dio_write_int16(AdvDocument* doc, adv_off_t offset, int16 val)`**

Writes the 16-bits **`int`**-type data into the position specified by **`offset`** in *Document* **`doc`**.

returned value		Size of written data
argument	**`doc:`**	*Document* for writing
	`offset:`	The writing position in *Raw Data*
	`val:`	The written data

◆ **`int32 adv_dio_write_int16v(AdvDocument* doc, adv_off_t offset, int num, const int16* val)`**

Writes the **`num`** parts of the 16-bits **`int`**-type data into the position specified by **`offset`** in *Document* **`doc`**.

returned value		Size of written data
argument	**`doc:`**	*Document* for writing
	`offset:`	The writing position in *Raw Data*
	`num:`	The number of data parts
	`val:`	The written data

◆ **`int32 adv_dio_write_int32(AdvDocument* doc, adv_off_t offset, int32 val)`**

Writes the 32-bits **`int`**-type data into the position specified by **`offset`** in *Document* **`doc`**.

returned value		Size of written data
argument	**`doc:`**	*Document* for writing
	`offset:`	The writing position in *Raw Data*
	`val:`	The written data

- **`int32 adv_dio_write_int32v(AdvDocument* doc, adv_off_t offset, int num, const int32* val)`**

  Writes the **`num`** parts of the 32-bits **`int`**-type data into the position specified by **`offset`** of *Document* **`doc`**.

returned value		Size of written data
argument	**`doc:`**	*Document* for writing
	`offset:`	The writing position in *Raw Data*
	`num:`	The number of data parts
	`val:`	The written data

- **`int32 adv_dio_write_int64(AdvDocument* doc, adv_off_t offset, int64 val)`**

  Writes the 64-bits **`int`**-type data into the position specified by **`offset`** in *Document* **`doc`**. In the computer environments where 64-bits **`int`**-type is not supported, **`int32`** will be used for the data type of **`val`** and the missing 32-bits part will be filled by **`0`**.

returned value		Size of written data
argument	**`doc:`**	*Document* for writing
	`offset:`	The writing position in *Raw Data*
	`val:`	The written data

- **`int32 adv_dio_write_int64v(AdvDocument* doc, adv_off_t offset, int num, const int64* val)`**

  Writes the **`num`** parts of the 64-bits **`int`**-type data into the position specified by **`offset`** in *Document* **`doc`**. In the computer environments where 64-bits **`int`**-type is not supported, **`int32*`** will be used for the data type of **`val`** and the missing 32-bits part will be filled by **`0`**.

returned value		Size of written data
argument	**`doc:`**	*Document* for writing
	`offset:`	The writing position in *Raw Data*
	`num:`	The number of data parts
	`val:`	The written data

◆ **`int32 adv_dio_write_float32(AdvDocument* doc, adv_off_t offset, float32 val)`**
Writes the **`float`**-type 32-bits data into the position specified by **`offset`** in *Document* **`doc`**.

returned value		Size of written data
argument	**`doc:`**	*Document* for writing
	`offset:`	The writing position in *Raw Data*
	`val:`	The written data

◆ **`int32 adv_dio_write_float32v(AdvDocument* doc, adv_off_t off-set, int num, const float32* val)`**
Writes the **`num`** parts of the 32-bits **`float`**-type data into the position specified by **`offset`** in *Document* **`doc`**.

returned value		Size of written data
argument	**`doc:`**	*Document* for writing
	`offset:`	The writing position in *Raw Data*
	`num:`	The number of data parts
	`val:`	The written data

◆ **`int32 adv_dio_write_float64(AdvDocument* doc, adv_off_t offset, float64 val)`**
Writes the 64-bits **`float`**-type data into the position specified by **`offset`** in *Document* **`doc`**.

returned value		Size of written data
argument	**`doc:`**	*Document* for writing
	`offset:`	The writing position in *Raw Data*
	`val:`	The written data

◆ **`int32 adv_dio_write_float64v(AdvDocument* doc, adv_off_t off-set, int num, const float64* val)`**
Writes the **`num`** parts of the 64-bits **`float`**-type data into the position specified by **`offset`** in *Document* **`doc`**.

returned value		Size of written data
argument	**`doc:`**	*Document* for writing
	`offset:`	The writing position in *Raw Data*
	`num:`	The number of data parts
	`val:`	The written data

7. Functions Related to AdvDatabox

◆ **`AdvDatabox* adv_dbox_new(void)`**
Opens **`AdvDatabox`**.

returned value		Pointer of opened **`AdvDatabox`**
argument		None

◆ **`bool adv_dbox_add(AdvDatabox* adb, const char* locator)`**
Stores the *Document* of the file specified by **`locator`** in the **`AdvDatabox`**.

returned value		Normal operation: a value other than **`0`** Error: **`0`** value
argument	**`adb:`**	**`AdvDatabox`** which contains **`AdvDocument`**
	`locator:`	The name of the file containing **`AdvDocument`**

◆ **`void adv_dbox_close(AdvDatabox* adb)`**
Closes **`AdvDatabox`**.

returned value		None
argument	**`adb:`**	Closed **`AdvDatabox`**

◆ **`AdvDocument* adv_dbox_find_by_documentid(AdvDatabox* adb, const char* docid)`**
Opens *Document* with *Document ID* defined by **`docid`** from **`AdvDatabox`**.

returned value		The pointer of corresponding **`AdvDocument`**
argument	**`adb:`**	**`AdvDatabox`** for search
	`locator:`	The string displaying *Document ID*

◆ **`AdvDocument* adv_dbox_find_by_property(AdvDatabox* adb, AdvDocument* prev, ...)`**
Searches in **`AdvDatabox`** for *Document* with specified *Property*. **`prev`** and **`“...”`** are the same as that of **`adv_dio_open_by_property`** (page 4).

returned value		The pointer of corresponding **`AdvDocument`**
argument	**`adb:`**	**`AdvDatabox`** for search
	`prev:`	*Document* matched with search conditions **`“...”`**
	`... :`	Search conditions

◆ **`int adv_dbox_count_by_property(AdvDatabox* adb, ...)`**
Counts the number of *Document*s in **`AdvDatabox`**, which matched with specified *Property*.

returned value		The number of corresponding **`AdvDocument`**
argument	**`adb:`**	**`AdvDatabox`** for search
	`... :`	Search conditions (uses the same setting procedure as **adv_dio_open_by_property**.

◆ **`AdvDocument* adv_dbox_open_nth(AdvDatabox* adb, int n)`**
Opens **`n`**th *Document* recorded in **`AdvDatabox`**.

returned value		The pointer of corresponding **`AdvDocument`**
argument	**`adb:`**	**`AdvDatabox`** for search
	`n:`	An integer

Example:

A part of the program that displays all *Document ID*s of *Document* included into **`adb`**.

```
void main(int argc,char* argv[]){

    int i=0;
    AdvDatabox *adb;
    AdvDocument *doc;

    adb = adv_dbox_new();
    adv_dbox_add(adb, "test.adv");
    while( (doc = adv_dbox_open_nth(adb, i++)) != NULL )
      fprintf(stderr,"%s¥n",adv_dio_get_documentid(doc));
 /*   adv_dio_get_documentid(doc) : returns Document ID of doc   */
```

Output results:

All *Document ID*s of *Document* included into **`adb`** are displayed.

```
6B8B4567:HDDM_FEGA@HDDM_Part[0]:190F:39B4FB9B
643C9869:HDDM_FEGA@HDDM_Part[0]:190F:39B4FB9B
74B0DC51:HDDM_FEGA@HDDM_Part[0]:190F:39B4FB9B
2AE8944A:HDDM_FEGA@HDDM_Part[0]:190F:39B4FB9B
238E1F29:HDDM_FEGA@HDDM_Part[0]:190F:39B4FB9B
3D1B58BA:HDDM_FEGA@HDDM_Part[0]:190F:39B4FB9B
2EB141F2:HDDM_FEGA@HDDM_Part[0]:190F:39B4FB9B
79E2A9E3:HDDM_FEGA@HDDM_Part[0]:190F:39B4FB9B
515F007C:HDDM_FEGA@HDDM_Part[0]:190F:39B4FB9B
12200854:HDDM_FEGA@HDDM_Part[0]:190F:39B4FB9B
216231B:HDDM_FEGA@HDDM_Part[0]:190F:39B4FB9B
1190CDE7:HDDM_FEGA@HDDM_Part[0]:190F:39B4FB9B
140E0F76:HDDM_FEGA@HDDM_Part[0]:190F:39B4FB9B
109CF92E:HDDM_FEGA@HDDM_Part[0]:190F:39B4FB9B
7FDCC233:HDDM_FEGA@HDDM_Part[0]:190F:39B4FB9B
41A7C4C9:HDDM_FEGA@HDDM_Part[0]:190F:39B4FB9B
4E6AFB66:DocumentList@HDDM_Part[0]:190F:39B4FB9B
```

8. Other Functions

◆ **`void adv_dio_copy_to_file(AdvDocFile* dfile, AdvDocument* doc)`**

Copies *Document* **`doc`** to *Adv* file indicated by **`dfile`**.

returned value		None
argument	**`dfile:`**	The target file *Adv* file for copy
	`doc:`	*Document* to be copied

◆ **`int adv_format_get_size(const char* format)`**

Returns the size of the data depending on the data format, which is indicated by the character string **`format`** used in *Property* to show the format of *Raw Data.* **`format`** is presented by combination of **`i1`**, **`i2`**, **`i4`**, **`i8`**, **`f4`** and **`f8`**.

returned value		The size of the data indicated by **`format`** (If non-permissible characters are found in the character string of **`format`**, **`-1`** will be returned)
argument	**`format:`**	A string of characters showing the format of *Raw Data*

Example:

```
int bytes1,bytes2;

bytes1 = adv_format_get_size("i4f8f8");
bytes2 = adv_format_get_size("int32");
printf("size of format = %d¥n",bytes1);
printf("size of int32 = %d¥n",bytes2);
```

Output results:

```
size of i4f8f8 = 20
size of int32 = -1
```

(**`-1`** is returned due to non-permissible characters "**`int32`**" in displaying the format of *Raw Data*)

◆ **`bool adv_format_pack(octet* buf, const char* format, ...)`**
Packs the data in **`octet`**-type **`buf`** according to **`format`**.

returned value		Normal operation: a value other than **`0`** Error: **`0`** value
argument	**`buf:`**	The location of stored data array (**`octet`**-type)
	`format:`	The string of characters which displays the format of *Raw Data*
	`... :`	The data sequence

◆ **`bool adv_format_pack_v(octet* buf, const char* format, va_list va)`**
Rearranges the data from the list of variable arguments **`va`** to fit the format **`format`** and packs it into **`octet`**-type **`buf`**.

returned value		Normal operation: a value other than **`0`** Error: **`0`** value
argument	**`buf:`**	The location of stored data array (**`octet`**-type)
	`format:`	The string of characters which displays the format of *Raw Data*
	`va:`	The list of variable arguments to present the data

◆ **`bool adv_format_unpack(octet* buf, const char* format, ...)`**
Unpacks the **`octet`**-type data from **`buf`** according to format **`format`**.

returned value		Normal operation: a value other than **`0`** Error: **`0`** value
argument	**`buf:`**	The packed data array
	`format:`	The string of characters which displays the format
	`... :`	The row of addresses of the variables which store the data

Index