CIFF

Specification on
Directory/File Organization
and
File Handling Protocol

Version : 1.0 revision 3
Date : February. 9, 1998

Revision history

CIFF Specification on Directory/ File Organization and File Handling Protocol, Feb 17,1997

Revision 2, December 24, 1997 (This document)

On page 1, official name of this file format is corrected to “Camera Image File Format”.

Revision 3, February 9, 1998 (This document)

On page 1, description about the 3 letter pre-fix of principal-image file names has been corrected.

Table of Contents

I. Purpose of the document

Among the regulations stipulated by the CIFF (Camera Image File Format), the present document describes those related to the names and storage positions for directories and files created on removable media, and those related to the methods to be used for their handling. For information on the internal formats of the individual image-data and audio-data files, refer to the separate document "CIFF Specification on Image Data File."

II. Directory/File name convention

The sub-directories holding the image, audio, and any other files generated by a camera should be of the form "CTG10100," that is, they should begin with the characters "CTG," and have the subsequent 5 characters form a base-ten integer, thereby resulting in an 8-character long name. The five-digit decimal integer is referred to as the directory number. Henceforth, this type of storage directory will be referred to as a "CTG directory."

CTG directories are positioned inside the "DC97" sub-directory, which is itself in the root directory.

When the fifth digit from the end of the directory number is a 0, an underscore ("_") is used instead. This is for ease of reading. For example, CTG00100 would become CTG_0100. Even if directories like CTG00100 (i.e., in which the fifth digit is 0) are present, the camera or application will ignore them.

Image- and audio-file names consist of eight characters, followed by three-character extensions. They take the form IMG10100.JPG or THM10100.JPG, that is, they consist of three characters(alphanumerics), followed by 5-digit decimal integers . The latter is referred to as the file number.

As in the case with directories, if the fifth digit from the end of the number in the file-name is a 0, an underscore ("_") is used instead. That is, IMG00100.JPG would become IMG_0100.JPG.

Image, audio, and other files having the same file number can be thought of conceptually as comprising a single image object. For example, the files IMG10100.JPG, THM10100.JPG, and SND10100.WAV are all files forming a single image object. In the following, such image objects will be referred to as "CIFF image objects."

For thumbnails, the first three alphanumeric characters used are fixed; they must be "THM." It is this prefix that is used to differentiate thumbnails from principal images. For the first three characters of principal-image file names, alphanumeric, "A" to "Z" or "0" to "9" shall be used. **The characters "THM," however, may not be used.**

The directory number (dirNo) of the directory in which a CIFF image object having a given file number (fileNo) should be located is determined uniquely according to the following formula:

dirNo = (fileNo-1) / kNumObjectInDirectory

kNumObjectsInDirectory is a constant indicating the number of CIFF image objects that should be stored in a single directory. We define this value as 50.

If this relationship is reserved, it can be seen conversely that the numbers of the CIFF image object files that should be present in directory number dirNo are those from dirNo * kNumObjectInDirectory +1 through dirNo * kNumObjectInDirectory + kNumObjectInDirectory.

For example, the file numbers of the CIFF image objects that can be present in directory number 0 are those from 1 through 50.

In **CIFF**, three files -- a principal-image file (extension JPG), its thumbnail file (extension JPG), and an audio file (extension WAV) -- comprise the standard files making up a CIFF image object.

The following is an example in accordance with the above specifications, indicating how a camera might generate directories and files.

```
root: \DC97
      \CTG_0000
            CTG_0000.CTG  (CTG file (described below))
            IMG_0001.JPG  (Image file-JPEG)
            THM_0001.JPG  (Thumbnail file-JPEG)
            SND_0001.WAV  (Audio File-WAV)
            IMG_0002.JPG
            ....
            IMG_0050.JPG
            THM_0050.JPG
      \CTG_0001
            CTG_0001.CTG
            IMG_0051.JPG
            THM_0051.JPG
            ....
      \CTG_0021
            CTG_0021.CTG
            IMG_1051.JPG
            THM_1051.JPG
            ....
      \CTG_0200
            CTG_0200.CTG
            IMG10001.JPG
            THM10001.JPG
            ...
      \CTG_2000
            CTG_2000.CTG
            IMG_0001.JPG
            ....
```

III. File-number Generation

It is desirable for each of the names of the files generated by a given camera to be unique. If the file names are unique, the directory names will also be unique, and it will therefore be more convenient for the user to manage his or her files. That is, the user will not have to worry about naming conflicts.

In view of this, in order to maintain the uniqueness of the file numbers, the camera maintains one file-number value continuously. The value so maintained is in the variable FileNoCurrent.

Normally, the value of FileNoCurrent will be 1 at the time the camera is shipped.

Whenever an image or an audio clip is recorded, the camera determines the file name by using FileNoCurrent as its file number, and with each item so recorded, the value of FileNoCurrent is incremented by 1.

In order to generate a CIFF image object having FileNoCurrent as its file number, a directory must be prepared in advance that satisfies the above-described relationship. In other words, if no CTG directory is present that has that directory number, it will be created at that time.

IV. Operations upon Insertion of Removable Media

The following procedure is followed when a removable medium is inserted. In the explanation below, the directory number corresponding to the FileNoCurrent value at the instant before the medium is inserted is referred to as DirNoCurrent.

1. When the camera senses the presence of a correctly operating piece of removable media, it treats that as the current drive. If an abnormality is detected, an error is displayed, and operations do not proceed.

2. When the camera first recognizes the drive (e.g., when a CF card is mounted, after batteries are replaced, etc.), it searches the active drive for a DC97 sub-directory under its root directory. If no such directory is present, the camera creates one. If there is a file, however, with the name DC97, the camera displays an error and does not proceed further.

3. The camera searches for CTG directories below the sub-DC97 directory. Any empty CTG directories are erased. The number of the CTG directory with the highest number found is saved, and the value is given the name DirNoFound.

4. If the file having the highest value for its file number is named according to the format for the naming of CTG directories, and is a file (not a directory), the value 1 greater than that file number is given to DirNoFound. If this results in a value above 99999, the camera displays an error and does not proceed further.

5. If no CTG directory is found, a CTG directory is created having the value of DirNoCurrent as its directory number.

6. If a CTG directory is present and the value of DirNoCurrent is greater than that of DirNoFound, a CTG directory is created having the value of DirNoCurrent as its directory number.

7. If a CTG directory is present and the value of DirNoCurrent is equal to or less than that of DirNoFound, the contents of the CTG directory numbered DirNoFound is searched, and the file having the largest file number is sought from among the files that have names containing file numbers that should be present there. This file's file number is assigned to the variable FileNoFound.

8. If the value of FileNoCurrent is less than that of FileNoFound, the value of FileNoCurrent is replaced by the value of FileNoFound + 1. This newly updated value of FileNoCurrent is used to re-determine the value of DirNoCurrent.

9. If at this point there is no CTG directory present that has DirNoCurrent as its directory number, a CTG directory is created having the value of DirNoCurrent as its directory number. This corresponds to the case in which the operation FileNoFound + 1 crosses beyond the directory boundary.

10. The values of FileNoCurrent and DirNoCurrent are determined according to the procedure outlined above. This results in these values being unique for the life of a single camera, as long as it is used in a normal manner.

V. Exception Processing for values near CTG99999

The last value that can be used as a directory number is 99999. Under normal use, it seems unlikely that CTG directories approaching this value would generally be produced due to the fact that it is unimaginable to take 5 million photos with a single camera.

Nonetheless, it is possible to cause this situation to arise artificially. That it, the user can create CTG directories manually. For situations such as this, the following processing exception is conducted:

- If the value of DirNoCurrent is 99900 or more, and not even a single CTG directory is present (and only in this case), the camera re-initializes the value of FileNoCurrent to 1.

- When a PC-host application discovers a piece of removable media containing a CTG directory with a value of 99900 or above, it urges the user to transfer all necessary files to backup, and then to delete all of the CTG directories.

The value 99900 is used because it is considered to provide some leeway before the value 99999. Thus, 99000 is acceptable as well.

By means implementing the above-described exception handling, it is assured that, if the camera is at least provided with a formatting function, the user -- him or herself -- can take action to bring about a recovery, even in a situation in which a PC is not available.

- As an additional option, even in situations in which the value of DirNoCurrent is 99900 or greater and any CTG directory is present, if that directory's number is below 99900, it is acceptable for the camera to re-initialize the file numbers and directory numbers corresponding to it.

VI. Other Notes

- **Uniqueness of File Numbers:** What is important is that all files numbers be unique; it is not necessary that they all be consecutive integers differing from their predecessors by 1. If for some camera-related reason a number is skipped occasionally, that poses no problem. Of course, if the numbers all become seemingly random, that will produce an unfavorable impression on the user, so this should be avoided.
- **Regarding Files Having Numbers Other Than Those Specified Above:** Even if files that have file names with numbers other than those assigned for administration within a given CTG directory are present, this will not effect the file name the camera generates next.
- **Handling of Files Having Various Extensions:** If files exist in a CTG directory that have names containing file numbers that should be administered within that directory, but with extensions that are not generated by that type of file (e.g., "DOC," "TXT," etc.), those files will be considered when the camera determines what file number it should generate next. That is, they will have to be considered as candidates for the value of FileNoFound. For instance, if XXX_0023.TXT is present and 23 is the highest file number in that directory, FileNoCurrent will be compared against the value 23.

VII. Regulations for the Manipulation of CIFF Image Objects

In this section we describe how a CIFF image object should be handled, based on its file names and storage location.

A. CIFF image object identification

A group of files that satisfies the following conditions is considered to make up a single CIFF image object:

- All the files are present in the same directory.
- The files follow CIFF-file naming conventions (i.e., XXXnNNNN.YYY, where n="_" or "1"-"9", N="0"-"9", and XXX and YYY are arbitrary strings [except that they must be selected from the 36 letters of the alphabet]).

- At least one of the files is among the standard files comprising a CIFF image object (i.e., has an extension of "JPG" or "WAV").

Even if the only files present are thumbnail files, or if only audio or only principal-image files are present, it is best if the group is still recognized as a CIFF image object. Depending on the player's capabilities, however, it is also acceptable to ignore such groups.

In cases in which there are multiple files that seem to represent the principal image, it is best if they are each considered to represent a separate CIFF image object. Depending on the player's capabilities, however, it is also acceptable to ignore them. For details on handling situations such as these, refer to the Appendix.

B. Copy

If the standard files comprising a CIFF image object[1] are present, they should all be copied to their destination together.

However, in situations such as when copying thumbnail images only, or principal images only, etc., in which the purpose of the copying being done is clear to the user, it is acceptable to transfer only those files that are required.

Even when there are files comprising a CIFF image object that are not comprehensible to the application doing the copying (e.g. *.DOC), it is still desirable that the application be able to copy them, as well. In particular, applications running on PCs, and dedicated playback units should have this capability.

C. Erase

All of the files comprising a CIFF image object should be deleted.

If even one of the multiple files making up a CIFF image object is protected, the entire object is considered to be protected. In this case, the fact that the CIFF image is protected should be indicated to the user, together with the information that the "Single Erase" cannot be conducted.

If as a result of deleting a file there are no files left in its parent directory, that directory itself is deleted.

If there is a file among those making up a CIFF image object that is not comprehensible to the replay unit, (e.g., "DOC," "TXT", etc.), it should be deleted only after requesting verification from the user as to whether it is permissible to do so. In situations in which it is not possible to request such verification, all of the other files comprising the CIFF image object to which that file might belong to should not be deleted.

---

[1] These files, having the same file number, include a principal image file XXXnNNNN.JPG, the corresponding thumbnail-image file THMnNNNN.JPG, and an audio file YYYnNNNN.WAV. Where XXX and YYY represent arbitrary character strings (with the provision that all of the characters must be among the 36 letters of the alphabet), and nNNNN is a string indicating the file number. Note, however, that it is preferable that the audio file's prefix of YYY be in keeping with the convention described in the Appendix.

D. Play back

A CIFF replay unit should be able to play back all of the standard files comprising a CIFF image object.

However, in the case in which play-back would require capabilities beyond the capacity of the replay unit, e.g., for audio files, extremely large principal-images, and other files (such as those having extensions that the unit does not understand), the unit should, at the very minimum, be able to inform the user of the presence of such files. If this is not possible, the files will not even be able to be deleted.

The performance, when a file is replayed of which the contents are not in accordance with the stipulations given in the separate document "CIFF Specification on Image Data File," may be defined in such a way that it is dependent on the capabilities of the individual replay unit. When replay is impossible, a message should be presented to the user indicating that the file is not a CIFF file.

VIII. CTG file

It is permissible to create a file (henceforth, CTG file) under a CTG directory having the same name as the directory, but with the extension "CTG" and any arbitrary contents, for use toward any private objective of the camera or the application. Note, however, that it is assumed that such files will be used only for temporary purposes.

If such files are used, the following points should be kept in mind:

- It must not be assumed that the file's contents will be preserved indefinitely. In other words, if such a file is used, then it is necessary for the application or camera to re-initialize the contents thereof every time the CTG directory is detected anew.
- If a CTG file is present and it has its read-only attribute set, the user will change that attribute and delete the file, before creating the CTG file once again. In other words, even if the file is given the read-only attribute, the probability of the file being deleted remains high.
- If all of the files except the *.CTG files end up being removed from the CTG directory, the *.CTG files will also be deleted. This is to prevent excessive empty space from being left over.

If files of this type are not used, their possible presence may be ignored. Alternatively, if they are nonetheless detected to exist, they may be deleted to save space.

IX. Appendix

A. CCIFFcollection

n the following, the procedure for determining whether the CIFF replay unit will recognize a given CIFF bject is given in a C++ type pseudocode. This section is included to prevent numerous different object-parsing ıethods from appearing for different types of vendors.

```
*
AN 30, 1997      Taku Yamagami    Canon Digital Imaging Group

ollowing is a psuedcode using C++language like notations describing how a
standard CIFF player understands CIFF image objects under the ROOT:\DC97, or
under arbitrary directories.

he class CCIFFcollection collects CIFF image objects located under given directory.
The CCIFFcollection object creates a list keeping CCIFFimageObject class object.
ach CCIFFimageObject class object keeps information of multiple files that
represents objects component such as thumbnail, full view, or sound annotation.

his psuedcode is focusing on how CCIFFcollection identifies individual object
and its component files. Describing detail of the List class or
CCIFFimageObject is out of scope.
he function like FindFirst(), FindNext() is well known and therefore
specific explanation is omitted.

his psuedcode is for demonstration/explanation purpose only. The author does NOT
guarantee this design works well on any platform.

/

////////////////////////////////////////////////////////////////
// class definitions

lass CCIFFimageObject

ublic:
     CCIFFimageObject();
     ~CCIFFimageObject();

     void ADDfile(DWORD dirNumber, LPCSTR filename,
            DWORD fileNumber, BYTE fileAttribute, DWORD fileTime);
       // add file information that belongs to a picture object
     void SetIndex(DWORD index);
     // set an index to be an identifier in the collection of picture object

rivate:
     //attributes
     BOOL m_protected;
     DWORD m_index;
     // an identifier used outside thie object. This object keeps the number
       // for the user.
     DWORD m_dirNumber;
     // could be (DWORD)(-1) indicating whether parent directory is CTGnNNNN or not.
     DWORD m_fileNumber;
     DWORD m_fileTime;
     // file time is the one for the full view file,
       // not one for the thumbnail or sound files.
     CLIST m_listOfFileComponent;
     // list of files belonging to this object.
     // i.e. full view(s), thumbnail view, sound(s)
     // Assume that CLIST object initialize itself in its constructor
```

```
lass CCIFFcollection

ublic
    void ScanDirectory(LPCSTR directoryPath, BOOL strictNumberChecking);
          // create picture collection for a given directory
    void ScanDC97directory(LPCSTR parentPath, BOOL strictChecking);
          // create picture collection for multiple CTGnNNNN directories
      // located under the given parentPath.

    CCIFFimageObject& GetLatest(DWORD posFromLatest);
    CCIFFimageObject& GetOldest(DWORD posFromOldest);
    virtual void EraseObject(CCIFFimageObject &object);
    virtual void PlayBackThumbnail(CCIFFimageObject &object);
    virtual void PlayBackFullView(CCIFFimageObject &object);

rotected:
    BOOL IsCIFFfullViewFileName(LPCSTR testFileName);
    virtual BOOL IsCIFFfileName(LPCSTR testFileName, DWORD *pFilNumber);
    CLSITofCIFFimageObject  m_CIFFobjectList; // list keeping CCIFFimageObject

    void AddMember( DWORD dirNumber, LPCSTR filename,
          DWORD fileNumber, BYTE fileAttribute, DWORD fileTime);


//////////////////////////////////////////////////////////////////
// implementation

OOL CCIFFcollection::IsCIFFfullViewFileName(LPCSTR testFileName)

    DWORD fileNumber;

    if(!IsCIFFfileName(testFileName, &fileNumber)){
          return FALSE;
    }

    if(testFileName has "THM" prefix){
          return FALSE;
    }


     // assume that this player understands JPG, CRW extensions
    switch (extension part of testFileName){
    case "JPG":  // At least “JPG” has to be supported
    case "CRW":  // “CRW” is an example of Canon proprietary file
          return TRUE;

    default:
          return FALSE;
    }


//////////////////////////////////////////////////////////////////
OOL CCIFFcollection::IsCIFFfileName(LPCSTR testFileName, DWORD *pFilNumber)

    if (length of filename part of the testFileName is NOT 8){
          return FALSE;
    }
    if (length of extension part of the testFileName is NOT 3){
          return FALSE;
    }
    if (last 5 character of filename part of testFileName does NOT
          represent valid file number){
          // 5 character must be nNNNN where n='_', or '1' thru '9', N='0' thru '9'.
          return FALSE;
    }
    *pFilNumber = convert the filename string to the file number based on
                        the file number convention;
```

```
   if(filename part of the testFileName has "THM" prefix){
       if (extension part of testFileName is NOT “JPG”){
            return FALSE; // file having "THM" prefix should be JPEG file
       }
   }
   return TRUE;



////////////////////////////////////////////////////////////
oid CCIFFcollection::ScanDirectory(
       LPCSTR directoryPath, BOOL strictNumberChecking)

    if(strictNumberChecking){
          currDirNumber = get directory number from the directory path;
          if ( currDirNumber is valid one,
                       i.e. if the directoryPath conforms to the CTGnNNNN convention){
                fileNumberMin = currDirNumber*50 + 1;
                fileNumberMax = currDirNumber*50 + 50;
          }else{
                return ;
          }
    }else{
          currDirNumber = -1;// indicating that parent directory may not be CTGnNNNN
    }

    result = FindFirst(directoryPath, &findData);
    for(;;){
          if (result == NOT_FOUND) break;

          if (findData.attribute indicates the file is a directory){
                continue;
          }
          if (!IsCIFFfileName(findData.filename, &fileNumber)){
                continue;
          }
          if ( strictNumberChecking &&
               ( (fileNumber < fileNumberMin) || ( fileNumberMax < fileNumber ) )
              ){
              continue;
          }
          AddMember(currDirNumber, findData.filename, fileNumber,
                findData.attribute, findData.time);

          result = FindNext(&findData);
    }


////////////////////////////////////////////////////////////////////
oid CCIFFcollection::ScanDC97directory(LPCSTR parentPath, BOOL strictChecking)
          // likely parentPath for CCIFFcollection is "ROOT:\DC97"

    result = FindFirst(parentPath, &findData);
    for(;;){
          if (result == NOT_FOUND) break;

          if (findData.attribute does NOT indicate directory){
                continue;
          }
          if (strictChecking &&
                (findData.filename does NOT conform to the CTG directory naming convention)){
                continue;
          }
          ScanDirectory(
                make directory path name concatenating parentPath and FindData.fileName,
                strictChecking);
```

```
            result = FindNext(&findData);
      }

      CheckFileParing();
      sort members of m_CIFFobjectList based on the modification time
            of the files belonging to each member object;


////////////////////////////////////////////////////////////////
oid CCIFFcollection::AddMember(
            DWORD dirNumber,
            LPCSTR filename,
            DWORD fileNumber,
            BYTE fileAttribute,
            DWORD fileTime,
            )

      CCIFFimageObject *pObject
            = in the m_CIFFobjectList, find an object having the filenumber
                  being same to the "fileNumber" argument;
      if (not found) {
            pObject = generate new CCIFFimageObject;
            add pObject to the m_CIFFobjectList;
      }
      //add file information to the pObject;
      pObject->ADDfile(dirNumber, filename, fileNumber, fileAttribute, fileTime);

      pObject->SetIndex(
            DWORD index = get current total number of items kept in m_CIFFobjectList
            /* This number is used to identify an picture object in a collection */);


////////////////////////////////////////////////////////////////
oid CCIFFcollection::CheckFileParing()
*
f a picture object has multiple full view files (On the PC side, this kind of situation is NOT very
are. User is likely to copy files to arbitrary directory.), this example separates them into
ndependent picture objects. This means it breaks thumbnail pairing even if there could be valid
are in the files. For instance, assume that an object has four image files
AAA_0001.JPG,BBB_0001.JPG, CCC_0001.JPG, THM_0001.JPG. All these files will be separated into an
ndependent object even if AAA_0001.JPG and THM_0001.JPG were valid pare. This behavior is the most
asiest way to resolve the problem and also serves end user's needs at minimum. i.e. It does not
gnore invalid pares and let user know its existence. We recommend this behavior as a minimum
equirement of the software running on the PC, or dedicated machine.

t is possible to improve the behavior by comparing file modification time of each files in order to
etrieve valid pare. This kind of improvement is left to each vender's effort.

he most important thing is NOT to ignore existence of those kind of files. We should provide
nformation to the end user as much as possible.

eanwhile, If a camera has to be a CIFF player, current model could be slightly heavy load to the
irmware. The camera's most important role is to take picture, NOT allowing user to edit directory
tructure. So, We recommend that camera can ignore such invalid paring. i.e. treat them as they are
ot existing. However, the ignored files should not be able to be deleted by the camera.

here could be an object that has only thumbnail or sound file. On this case, at least on the PC
ide or on a dedicated machine, the player should display the thumbnail to the end user and let her
r him to know the absence of the full view.

here could be an object that has unknown files which might be the full view file. For instance,
CRW” is the Canon’s proprietary full view file extension that would be not understandable to other
ender’s player. On that kind of case, player should inform user that there is a unknown file that
eems to be a full view file.
/

      for(;;){
            pObject = get an object in the m_CIFFobjectList;
```

```
        // this increment object pointer to next object,
        // so that subsequent "get" gets next object.
        if (not found) return;

        if (the pObject has multiple full view file component){
               // e.g. AAA_0001.JPG, BBB_0001.CRW, CCC_0001.TIF
               // On this case, treat each individuals as an independent picture object.
               // So, remove the pObject from the list and spawn CCIFFimageObjects
               //           having each full view file.
               // In order to check if a file name conforms to CIFF full view
               //     file naming convention, use the IsCIFFfullViewFileName() funciton.

               remove the pObject from the m_ListDEobject;
               CCIFFimageObject * pNewObject;
               for (each file belonging to the pObject) {
                      pNewObject = generate new CCIFFimageObject;
                      pNewObject->ADDfile(get the file information from the pObject);
                      add pNewObject to the m_CIFFobjectList;
               }
               continue;
        }
        if (the pObject has multiple wave file component){
               // e.g. AAA_0001.WAV and BBB_0001.WAV
               // On this case, treat each individuals as an independent orphan sound, i.e.
               // the sound having no owner.
               // So, remove the pObject from the list and spawn CCIFFimageObjects
               // having each sound file.

               for (each sound file belonging to the pObject) {
                      remove the sound file form the file list of pObject;
                      pNewObject = generate new CCIFFimageObject;
                      pNewObject->ADDfile(get the sound file information from the pObject);
                      add pNewObject to the m_CIFFobjectList;
               }
               continue;
        }
        // Here, pObject should NOT have multiple thumbnail file.
        // So, there is no check on that case.

        if (any of the file components of the pObject has
                    SYSTEM, VOLUME, or HIDDEN  attribute){
               remove the pObject from the m_ListDEobject and free it;
               continue;
        }

        if (any of the file components of the pObject has READONLY attribute){
               pObject.m_protected = TRUE;
        }else{
               pObject.m_protected = FALSE;
        }
    }


////////////////////////////////////////////////////////////////////
CIFFimageObject CCIFFcollection::GetLatest(DWORD posFromLatest)

    return "posFromLatest"-th member positioned from latest member in m_CIFFobjectList;


////////////////////////////////////////////////////////////////////
CIFFimageObject CCIFFcollection::GetOldest(DWORD posFromOldest)

    return "posFromOldest"-th member positioned from oldest member in m_CIFFobjectList;


////////////////////////////////////////////////////////////////////
oid CCIFFcollection::EraseObject(CCIFFimageObject &object)
```

```
   if (object.m_protected){
       return; // should not delete protected object
   }
```

Delete files belonging to the object.

If the object has unknown files that could be full view file, or some related files to the object, (e.g. files having CRW, TIF extension might be some vender unique full view file, files having TXT, or DOC might be some related document to the image) let end user know its existence and get approval from the user to delete it. If player can not inform user that the unknown file is existing, it should not delete the CIFF object.

If corresponding directory become empty, i.e. the directory does not have any files, delete the directory as well.

Remove the object from the m_CIFFobjectList.

```
//////////////////////////////////////////////////////////////////
oid CCIFFcollection::PlayBackThumbnail(CCIFFimageObject &object)

   if (object has thun\mbnail file){
     play back it;
   }else{
     tell user that there is not thumbnail for the object;
   }


//////////////////////////////////////////////////////////////////
oid CCIFFcollection::PlayBackfullView (CCIFFimageObject &object)
```

if object has a full view file, play back it.
If not, tell user that full view is missing.
Furthermore, if the object has unknown files that could be full view file, or some related files to the object, (e.g. files having CRW, TIF extension might be some vender unique full view file, files having TXT, or DOC might be some related document to the image) let end user know its existence.

B. Convention for Audio-file Filename Prefixes

The following convention was established for audio-file filename prefixes (i.e., for the first 3 characters), in order to permit the recording period to be deduced from the filename and file size alone. With this convention, the audio clip's length can be calculated without examining the contents of the file itself. Of course, it is necessary to parse the file's header when the file is to be replayed.

	Coding method	Sampling frequency
“SND”	8 bit/sample PCM	11.1 k Hz
“SNC”	4bit/sample ADPCM	11.1 k Hz
“SHD”	8 bit/sample PCM	22.2 k Hz
“SHC”	4bit/sample ADPCM	22.2 k Hz

Note: "AUT," "CRW," "MED," "SML," "TIM," and "TM2" were used by the Canon PowerShot600 Digital Camera. These all have specifications identical to those of "SND" in the explanation above.