COFOps Software User's Manual
Version 1.5 of 1998/06/04 06:45:26

Rick Crowhurst
rick@picturel.com

Picture Elements, Inc.

1. Scope

This document describes the **COFOps** software package, which is a TCL-based tool set for creating, manipulating, and verifying COF file sets.

2. Introduction

COF (Common Output Format) is a standard for the interchange of images and associated financial data between the Federal Reserve Bank and financial institutions; it describes how this information is organized into sets of files, with particular names and formats, on a piece of COF media. **COFOps** is a public domain package of tools for creating, manipulating, and verifying COF files and files sets. Its purpose is to provide standard test file sets for testing COF readers, and to verify file sets in order to test COF writers.

COFOps is provided in source code; it is implemented in the C and TCL languages. It was developed and tested under the Linux operating system, but it should be easily transportable to any platform that supports C and TCL. The use of TCL (Tool Control Language) allows the manipulation tools to be used interactively as well as within TCL scripts; TCL is described in Tcl and the Tk Toolkit by John Ousterhout. It is a standard component of Linux; for platforms that don't provide TCL, it can be downloaded as source code free of cost by anonymous FTP from

```
ftp://ftp.smli.com/pub/tcl
```

Section 3 of this document describes the management of the software package, including unpacking, program generation, and program installation. Section 4 describes the executable programs provided by the package, and section 5 describes the commands provided in the TCL extensions that allow COF files to be manipulated from within TCL scripts or interactively. Finally, section 6 lists a number of limitations of the current version of the package.

3. COFOps Package Management

This section gives Linux command sequences for performing various routine manipulations of the software package. In all cases, it is assumed that the directory ~ has been chosen as the home directory for the package.

3.1. Unpacking the Software

This software is distributed as the gzip'd tar file "`cofops-X.XX.tgz`", where "`X.XX`" gives the version number. To unpack the distribution file, move it into ~ and execute:

```
cd ~
gunzip cofops-X.XX.tgz
tar xf cofops-X.XX.tar
```

It is convenient, but not required, to add a symbolic link `cofops`, which conventionally refers to the current version of **COFOps**:

```
cd ~
ln -s cofops-X.XX cofops
```

This link will be assumed in the sequel.

3.2. Generating the Libraries:

After the package has been unpacked, the libraries can be generated by:

```
cd ~/cofops
make
```

This will produce the TCL extensions in the form of the shared libraries `libini.so`, `libimg.so`, and `libdbf.so`.

3.3. Installing the Package

After the libraries have been generated, the package can be installed with:

```
cd ~/cofops
su
make install
```

This will copy the libraries into `/usr/lib` and the program `cofck.tcl` into `/usr/local/bin` as `/usr/local/bin/cofck`. The package is now ready for use.

3.4. Document Generation

To generate this document, execute:

```
cd ~/cofops
make cofops.ps
```

4. COFOps Programs

This section describes the executable programs provided by **COFOps**. These programs are implemented as TCL scripts and installed in `/usr/local/bin` by "`make install`", as described in section 3.3. They require that the **COFOps** libraries be installed in `/usr/lib` (this is also done by "`make install`") and that the TCL shell `/usr/bin/tclsh` be present.

4.1. cofck

cofck is a program that checks the validity of COF file sets and reports any deviations from the COF specification (version 1.3) that are discovered. It takes a single argument which is the directory in which the file set is found. For example, in Linux, the command

```
cofck /mnt/cdrom
```

checks the COF file set on a (mounted) CD-ROM. The program displays "WARNING" for variances from the specification that do not force **cofck** to abort the check; for fatal errors, it displays "ERROR".

4.2. cofgen

cofgen will be a program that generates a canonical COF data set. Currently, **cofgen** is unimplemented.

4.3. cofsum

cofsum computes and returns the COF checksum of any file, or of the standard input if no file is given.

5. COFOps TCL Extensions

TCL allows the programmer to extend the TCL language by adding C functions that can be called by the TCL interpreter. In Linux, these "TCL extensions" can be packaged as shared libraries and loaded dynamically. In **COFOps**, the low-level support for the programs described above in section 4 is provided by the TCL extension libraries `libini.so`, `libimg.so`, and `libdbf.so`, and these extensions are also available to the user for use either in TCL scripts or interactively with the TCL shell; they allow individual COF files to be created, examined, and manipulated.

Each TCL extension provides commands associated with a particular COF file format: `libini.so` handles header and trailer files, which are in ".INI" format, `libimg.so` is for the .IMG files, and `libdbf.so` is for the data (.DBF) files. A typical application will start by loading the extensions with:

```
load libini.so
load libimg.so
load libdbf.so
```

Since the TCL `load` command looks in `/usr/lib` by default, the full path names need not be provided.

The following sections describe each extension in detail.

5.1. TCL Extension for COF Header and Trailer Files (`libini.so`)

This extension supports the reading of files in Windows ".INI" format, which includes all COF header and trailer files. Note that only READ support is provided; modifying .INI files in TCL is more appropriately done by invoking a text editor.

This extension provides the following command:

`iniRead` *array path*

> This command opens an parses a .INI file at the given *path* and builds the named *array*, which holds the following elements:
>
> $*array*`(.)`
>
> > This element is a TCL list containing the names of all of the sections found in the .INI file
>
> $*array*`(`*section*`)`
>
> > This element is a TCL list containing the names of all of the variables assigned in all of the sections named *section*.
>
> $*array*`(`*section*`,`*variable*`)`
>
> > This element is the string value assigned to the given *variable* in the given *section* of the .INI file

For example, if the file `test.ini` holds:

```
[FileName]
FileName=test.ini

[Info]
SomeNumber=6
SomeString=This is a string
```

and the following TCL commands are executed:

```
load libini.so
iniRead ini test.ini
puts $ini(Info,SomeString)
```

then the following output is produced:

```
This is a string
```

As another example, the following TCL sequence parses a .INI file into an array, then displays the array in .INI format:

```
load libini.so
iniRead ini test.ini
foreach section $ini(.) {
    puts "\[$section\]"
    foreach variable $ini($section) {
        puts "$variable=$ini($section,$variable)" } }
```

5.2. TCL Extension for COF Image Files (`libimg.so`)

This extension supports the parsing and interpretation of COF ".IMG" files by providing the following TCL commands:

`imgOpen` *name path*
: This command opens and interprets the file at the specified *path* as a .IMG file, and creates the given *name* as an access function for the file; *name* can then be used as described by the following three commands. If any syntax errors are found, the function returns with an error message.

name `items`
: This command returns a TCL list that describes the entire parsed file; its format is described below.

name `verification`
: This returns two numbers: the size of the file in bytes and the calculated checksum.

name `extract` *file itemid viewid*
: This command extracts a selected image from the .IMG file and writes it into the given *file*, which must be open for BINARY writing using the TCL `open` command. *itemid* and *viewid* select the image; they are derived from the items list.

The items list returned by the "*name* `items`" command is illustrated by the TCL example below. It has an element for each item in the .IMG file; each such element is itself a list whose elements are the item id, the item data list, and the item view list. Each view in the view list is a list that gives the view id, the side (F or B), the type (TIFFG4, etc), the offset within the image file, and the length of the view within the image file.

```
load libimg.so
imgOpen items test.img
foreach item $items {
    set item_id [lindex $item 0]
    set datalist [lindex $item 1]
    set views [lindex $item 2]
    foreach view $views {
        set vid  [lindex $view 0]
        set side [lindex $view 1]
        set type [lindex $view 2]
        set foff [lindex $view 3]
        set flen [lindex $view 4] } }
```

5.3. TCL Extension for COF Data Files (`libdbf.so`)

This extension supports the reading and writing of COF data (dBASE) files by providing the following TCL commands:

`dbfCreate` *path field-spec ...*
: This command creates a .DBF file with the given *path* and with fields specified by the given *field-spec*s. *field-spec* is described below.

`dbfOpen` *name path*
: This command opens the file with the given *path* and creates the given *name* as an access function for the file; *name* can then be used as described by the following three commands.

name `nrecords`
: Returns the number of records in the .DBF file.

name `headerSize`
: Returns the size in bytes of the .DBF file header.

name `recordSize`
: Returns the size in bytes of a .DBF file record.

name `fields`
: This command returns a list of the fields of the .DBF file. Each field is represented in the *field-spec* format, described below.

name `add` *field-value ...*
: This command adds a new record with the given *field-value*s to the .DBF file. The quanity and contents of the fields are checked. Each *field-value* is represented in the format appropriate to the field type: for character data, the *field-value* is a string with trailing blanks removed; Numerical data is given as a TCL number.

name `record` *number*
: This command returns the record with the given *number* if it exists and is not marked as deleted; otherwise, it returns with an error.

The *field-spec* argument used by the commands above has the following definition:

field-spec
: A field specification is a TCL list with the following elements:

name	the name of the field
type	the field type, either C, D, L, or N
length	the length of the field
decimal	the size of the decimal part

6. Limitations of the Current Version

The current version of **COFOps** has the following limitations:

1. Generation of COF file sets is not implemented.
2. Manipulation and verification of COF index (.NDX) files is not implemented.
3. Handling of COF media sets consisting of than one piece of media is not implemented.