
ViXeN Documentation

Release 1.0rc3.dev0

Prabhu Ramachandran and Kadambari Devarajan

May 23, 2018

1	ViXeN: View eXtract aNnotate media data	3
1.1	Features	3
1.2	History	4
1.3	Support	4
1.4	Citing ViXeN	4
1.5	Credits	4
1.6	Changelog	4
2	Installation and getting started	7
2.1	Dependencies	7
2.2	Installing ViXeN on GNU/Linux	8
2.3	Installing ViXeN on Mac OS X	8
2.4	Installing ViXeN on Windows	9
2.5	Installing using <i>pip</i>	9
2.6	Troubleshooting	9
3	Using ViXeN	11
3.1	Setting up a project	12
3.2	Viewing media	13
3.3	Keyboard shortcuts for navigation	14
3.4	Searching	14
3.5	Exporting the tag information to a CSV file	15
3.6	Importing tag information from a CSV file	15
3.7	Processing media files	15
4	Indices and tables	19

ViXeN is a simple tool to facilitate easily viewing, adding, and annotating metadata associated with media. ViXeN has been designed primarily to assist field biologists with managing the large amount of media they collect in a minimally intrusive manner.

ViXeN: View eXtract aNnotate media data

ViXeN is a simple tool to facilitate easily viewing, adding, and annotating metadata associated with media. ViXeN has been designed primarily to assist field biologists with managing the large amount of media they collect in a minimally intrusive manner.

One may think of ViXeN as a special, customizable file browser with which one may view and edit metadata associated with media files like videos, images, and audio.

ViXeN is open source and distributed under the liberal [BSD license](#).

1.1 Features

- Works with different types of media, videos, images, audio, text, and PDF.
- Supports any number of user-defined metadata fields per project.
- Does not modify the original media but keeps its metadata separately.
- Powerful searching through metadata.
- Lightweight and easy to install. No server setup required.
- Cross-platform: works on Linux, OS X, and Windows.
- Simple browser-based UI.
- Support to export metadata to a CSV file and import tags from a CSV file.
- Ability to add metadata for media through an external program or a Python script.
- Support to allow user-defined conversions of media to supported versions.
- Open source.

ViXeN has currently been tested with about 350k files in a single project. It will work fine for larger projects but will be progressively slower the larger the number of files. The speed does not however depend on the nature of the media. ViXeN does not currently support multiple people working on the same project at the same time.

1.2 History

ViXeN was envisaged as a ‘no-fuss’ means to manage videos from 32 camera traps that were set up in the Banni grasslands in north-west India to survey species occurrence and interaction patterns, with a focus on carnivores. The research project began in 2013 and as of 2015 has resulted in 6000+ videos. When confronted with the daunting task of viewing and managing such volumes of media files, we realized that there was a dearth for tools to aid in the processing of such data. There were data managers for images but nothing suitable for video (and to a certain extent, audio) files.

1.3 Support

If you have any questions or are having any problems with ViXeN, please email or post your questions on the vixen-users mailing list here: <https://groups.google.com/d/forum/vixen>

The ViXeN issue tracker and source code are available at: <https://github.com/vixen-project/vixen>

1.4 Citing ViXeN

You may use the following article to formally cite ViXeN if you wish:

- Prabhu Ramachandran and Kadambari Devarajan, *ViXeN: An open-source package for managing multimedia data*, *Methods in Ecology and Evolution*, 2041-210X, 2017.

1.5 Credits

Designed and developed by [Prabhu Ramachandran](#) and [Kadambari Devarajan](#).

Many thanks to [Kamal Morjal](#) for the CSS layout of the UI and [Khushalsingh Rajput](#) who implemented the ViXeN logo based on Kadambari’s concept and design.

1.6 Changelog

1.6.1 1.0rc2

- Release date: May 4, 2018.
- Prepare for release on PyPI.
- Add pause/stop buttons when processing files and clean up processing UI.
- Fix bug with saving projects which had tagger processors.
- Support to copy a project.
- Awesome new logo for the application and documentation.
- Improved the rescan functionality to warn users about what it does and remove dead files from the database.
- Add a new “text” tag type for larger text input.
- Warn user when they add unsupported tag names (names with spaces or those starting with an _).

- Add button to re-order the tags.
- Fix UI slowness when editing a lot of text on firefox.
- Fix an issue with CSV export with unicode text.
- Support for Python 3.x.

1.6.2 1.0rc1

- Release date: Mar 17, 2017.
- Massive performance improvements (> 10x) for handling many files. This makes it possible to comfortably handle many more files than before.
- Documentation improvements.
- Small UI improvements for search and navigation.
- Fix bug when starting vixen for the first time – if one added a project it would not show correctly until vixen was re-launched.
- Fix issue with notifications on firefox.
- Add ability to view the application log, to help with reporting issues.
- Notify users of any unexpected errors so these can be reported.
- Improved test coverage.

1.6.3 0.9

- Release data: Jan 18, 2017.
- First public release.
- Easy to install binaries on Linux, OS X, and Windows.
- Basic documentation.

Installation and getting started

To install ViXeN, you first need to know the platform you will be running it on. ViXeN binaries are available for Linux (64 bit), Mac (64 bit), and Windows (32 and 64 bit). You can install ViXeN in a Python environment easily too but this is not necessary if you do not use Python.

In order to use ViXeN you will need to have a working browser. See the *Dependencies* section for more information on the dependencies. Installing ViXeN is relatively straight-forward. Please see the instructions for installing ViXeN in the relevant sections below.

- *Dependencies*
- *Installing ViXeN on GNU/Linux*
- *Installing ViXeN on Mac OS X*
- *Installing ViXeN on Windows*
- *Installing using pip*
- *Troubleshooting*

2.1 Dependencies

2.1.1 Core dependencies

Please make sure you have a recent, functional browser such as: - Mozilla [Firefox](#) (any recent version) - Google [Chrome](#) (any recent version) - Internet Explorer (IE 9 and above)

ViXeN is tested to work best on [Firefox](#) and [Chrome](#).

2.1.2 Optional dependencies

The optional dependencies are:

- `ffmpeg`

This is only needed if you wish to use `ffmpeg` to convert any of your video media.

2.2 Installing ViXeN on GNU/Linux

On GNU/Linux systems one may install the dependencies by following three very simple steps:

- Download the Linux binary from the [ViXeN releases](#).
- Unpack the `vixen*.tgz` file.
- Run the application inside the unpacked directory.

More detailed instructions are given below.

2.2.1 Using Nautilus

1. Download the ViXeN binary file (say `vixen-1.0-linux64.tgz`) from [ViXeN releases](#) to your preferred directory.
2. Right click and select 'Extract Here' or 'Open With Archive Manager'. A ViXeN folder will appear in the directory you have selected (in this case the folder will be `vixen-1.0`).
3. Open the directory and double click the ViXeN file (the file is `ViXeN.desktop`). The ViXeN application will open on the browser.
4. If the above did not work you can try to run the `vixen` file. This will run **if** Nautilus supports running executables. If it does not, run the command shown in *Using the Command Line*.

2.2.2 Using the Command Line

After downloading and unpacking ViXeN either *Using Nautilus*, run the following commands on the terminal:

```
$ cd vixen-1.0
```

Where `vixen-1.0` is the extracted ViXeN binary. Suppose you downloaded and unpacked the `vixen-1.0-linux64.tgz` file in a folder named `Software` in your home directory, then this command will be:

```
$ cd Software/vixen-1.0
```

Then run the application using:

```
$ ./vixen
```

2.3 Installing ViXeN on Mac OS X

1. On OS X, download the ZIP file from the [ViXeN releases](#) page.
2. Unpack it.
3. Run the resulting `ViXeN.dmg` file by double clicking it.

You may move this dmg file anywhere you like or move it to your `Applications` folder if you wish to.

2.4 Installing ViXeN on Windows

1. On Windows, download the ZIP file for your platform from [ViXeN releases](#).
2. Unpack the ZIP file.
3. Inside the unpacked directory is a `vixen.bat` script or a `vixen.lnk` shortcut that you can use to run the application.

The application will open in your default browser window.

2.5 Installing using *pip*

If you have no idea what `pip` is, then you should look above for suitable instructions. If you are familiar with Python and wish to install ViXeN in your own Python environment, you can simply run:

```
$ pip install vixen
```

The requirements for ViXeN are fairly modest.

Otherwise, if you wish to install the development version, you can clone the ViXeN git repository at <https://github.com/vixen-project/vixen> and then do the following:

```
$ cd vixen
$ pip install -r requirements.txt
$ python setup.py install
```

If you want to stay up-to-date and not have to re-install each time you update the git repository, you could instead do:

```
$ python setup.py develop
```

Once installed you can start the application by:

```
$ vixen
```

To see any command line options try this:

```
$ vixen -h
```

That's about it.

2.6 Troubleshooting

If the application fails to run or you have any problems, please look at the log file `vixen.log` located inside the `.vixen` folder in your home directory.

- On Linux this is typically in the directory `/home/username/.vixen`.
- On OS X this is in `/Users/username/.vixen`.
- On Windows this may be `C:\Users\username\.vixen`.

Email your log file to the developers or the mailing list.

The ViXeN mailing list is available at <https://groups.google.com/d/forum/vixen>

This is a simple tutorial on how to get started using ViXeN and its features.

When you first start the application it should open a new page on your default browser. If this is not a supported browser, you can simply copy the URL on the location bar onto a supported browser such as Chrome or Firefox and the UI should load correctly.

When you first start you will have a rather empty page with an button on the left panel to create a new project as seen in the figure below:



Fig. 1: Start page when ViXeN is started for the first time.

In the figure, on the top right is a button to stop the ViXeN application. Clicking on this will close ViXeN and one

may safely close the browser tab. The top left has three links, “Home”, “Help”, and “Log” the home link brings the user to the page where one can edit or view the different projects. The help link takes a user to the documentation. If you are using a binary, this will take one to a copy of the documentation bundled with the binary. The “Log” shows the vixen application logs in a separate tab. This is handy if you are running into errors and is useful when you are reporting problems.

3.1 Setting up a project

Create a new project by clicking on “New project”. Fill in the fields of the project on the right side. An example of this can be seen in the figure below.

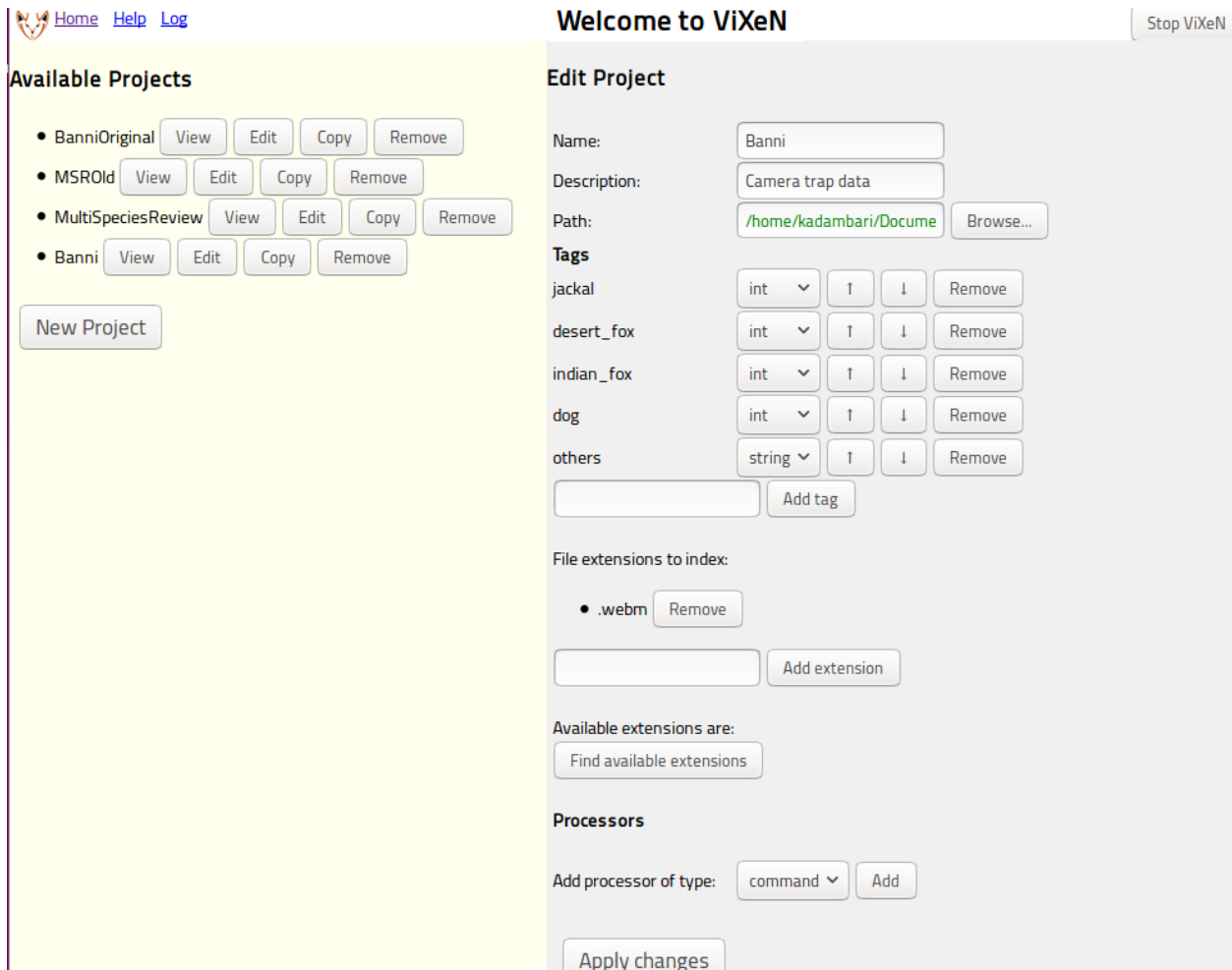


Fig. 2: Adding a new project and setting it up.

The important fields to fill are:

- Name: set this to a suitable name for your project.
- Path: this is the path to the root of your media files that you wish to index. Click on “Browse...” to choose the directory from a file browser. You may also directly type the directory path on the text box. All files inside this directory can be “indexed”. You may choose to index only specific extensions by adding specific extensions on the field below the “Tags” field.

- **Tags:** this is a very important field. These define the various metadata tags associated with your media. You may add as many fields as you desire. A tag can be either a string, integer, float, or boolean. For each media file, you will be able to change/edit these fields when viewing the project. You can add multiple tags in one go by separating them with commas, for example: fox, jackal, dog. Once added you can change the type of the tag on the UI. You can add and remove tags later on also. One default tag is always added called “completed” you may remove it without any loss of functionality if you do not need it.
- **File extensions to index:** this defaults to all extensions, you may specify any extensions you specifically wish to index. For example specifying `.png`, `.jpg` and clicking on the “Add extension” will index only the png and jpg files. The button, “Find available extensions” will show you a list of all extensions inside the specified path.
- **Processors:** You may add a variety of processors that allow you to either convert your media, copy your media, add tags using Python scripts, or use an external program to add tags. This is discussed in greater detail in the section *Processing media files* below. Processors are entirely optional.

Once you have setup the project, simply click on “Apply changes” for ViXeN to quickly scan all the files and make its internal database. Depending on the size of your directory, this should take a few seconds. Once this is done, you can click on the “View” button on the left pane to view the media.

If you do not need a project anymore, you may simply click on the “Remove” button remove the project. This will remove only the metadata, your media will be untouched.

Note that when the files are indexed the following tags are always available:

- **ctime:** date: time of most recent metadata change on Unix, or the time of creation on Windows.
- **mtime:** date: time of most recent content modification.
- **path:** string: the full path of the file.
- **relpath:** string: the relative path to the file with respect to the project root.
- **size:** int: the file size in bytes.
- **type:** string: the type of the file (video, audio, image, html, pdf, etc.)

3.2 Viewing media

The view interface is very simple and divided into two parts. An example view is provided in the following figure.

On the left side you will see a simple directory browser. Clicking on a directory (shown typically in bold with a trailing `/`) will navigate into this directory and clicking on a file will display the media on the right side of the page. Below the directory browser, the metadata tags of the media file are shown. One may edit the tags as one sees fit.

On the right side, the media file is shown. Any file format that the browser can render is typically shown currently this works for videos (webm, ogg theora video), image files (`.png`, `.jpg`, `.gif`, `.svg` etc.), audio files (`.mp3`, `.m4a`, `.ogg`, etc.), HTML, text, and PDF. The rendering is really dependent on the browser and your platform.

If the browser does not support the media but your operating system does, you can open the file with the system viewer by clicking on the “Name” field of the media on the left panel. This is typically shown as a blue link. This will open the file using the operating system’s registered viewer for the particular file.

ViXeN thus makes it easy to view the data on the right and update the metadata for each file.

Note: It is important to remember to save the project after changing the metadata. This can be done by pressing the “Save” button or pressing “Command+S” or “Control+S”.

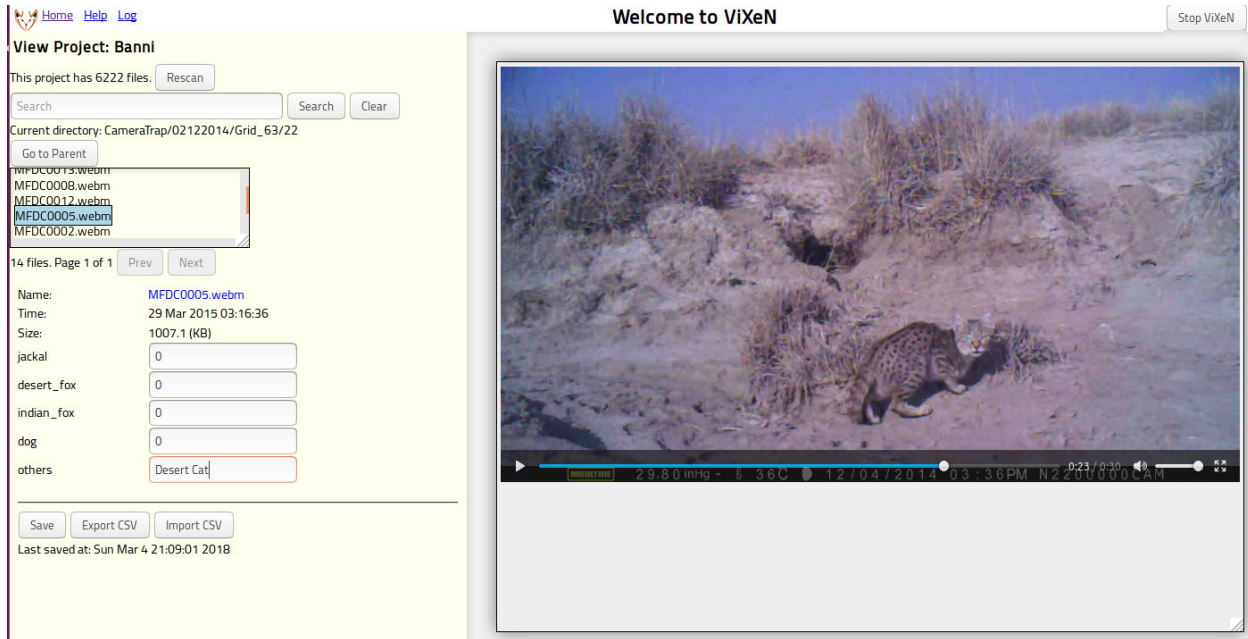


Fig. 3: Viewing a project on ViXeN.

3.3 Keyboard shortcuts for navigation

While navigating the directory browser, there are a few useful keyboard shortcuts:

- Pressing “h” or the left arrow key will go to the parent directory.
- “k”, “n”, or the “down” arrow key will go to the next file,
- “j”, “p”, or “up” arrow will go to the previous file/directory,
- “l”, “enter”, or the right arrow key will either select/view the file or navigate into a sub-directory.
- “Command+s” or “Control+s” will save the project.

3.4 Searching

One powerful feature with ViXeN is the ability to search through the metadata.

By default searching for a string in the search box and pressing return/enter or pressing the search button will search for the occurrence of the string in the full path of the media file.

To search for specific tags, let us consider an example project with the metadata tags “fox” (an integer), “jackal” (an integer), and “others” (string).

- To find all the media which have a single fox, one types: `fox:1`
- To find all the media which greater than one fox, one types: `fox:>1`
- To find all the media which greater than one fox or one jackal, one types: `fox:>1 OR jackal:1`
- To find all the media where the “others” tag has a gerbil one types: `others:gerbil`.
- To find all the media where there is a gerbil and a single jackal one types: `jackal:1 AND others:gerbil`

- To find the occurrence of the text “desert cat” in the “others” tag one types: `others:'desert cat'`. Note the use of the single quotes.

In addition, one may also search by the time of the media. Each media file’s creation time (`ctime`) and modified time (`mtime`) are also indexed automatically. One can search for the time as follows:

- for all images modified in 2015: `mtime:2015`,
- for all images modified in 2015 January: `mtime:201501` or `mtime:'jan 2015'`.

ViXeN uses [whoosh](#) to parse the query string. For more details on the query language see the [date parsing documentation](#).

3.5 Exporting the tag information to a CSV file

Once the tags have been entered one can export the metadata to a CSV file. Simply click on the “Export CSV” button and you will be prompted for a file. This file will contain all the tags for the data.

3.6 Importing tag information from a CSV file

One may also import tag information from a CSV file. Click on the “Import CSV” button, supply a file and it will import the tags. The CSV file must have a “path” column which should be exactly the same path as the corresponding media file. If there is a doubt as to what path is stored by Vixen, export the project data to CSV and look at the path column.

It is important to note that only tags that have already been defined in the project will be imported. The column name of the CSV file should match the tag name exactly. Any columns which do not have corresponding tags will not be imported.

Finally, after importing the tags, one must save the project to have the changes be stored to disk.

3.7 Processing media files

One can process media files in a variety of ways. When editing a project, one can add processors. The following processors are available.

- Command: run a user-defined command on the media files. Useful for format conversions.
- Tagger: run a user-defined program to tag the media.
- Python: call a user-defined Python function to do whatever desired.

The processors are described in greater detail below. One can add a processor and test it on a single file to see if it works correctly and then process the entire set of files. Always remember to save the project after the processing has completed and the results are satisfactory.

Note that if you have any search results and then run the processing, it will run the processing only on the searched files.

When running the processing, the UI will present a button to pause the execution of the processing or to stop it entirely. Once a file has been processed by a particular processor it will not be processed again. If you wish to re-do the processing for the already processed files, you will need to remove the processor and add it again by editing the project.

3.7.1 The Command processor

This processor allows the user to run an arbitrary command on the media. This is typically used to perform file format conversions. For example, if one wishes to convert an unsupported video with the extension `.avi` file to a `.webm` file, one may use `ffmpeg`. To run `ffmpeg` on the media, one simply adds a command processor and sets the “command” field to: `ffmpeg -i $input $output` and set the input extension field to `.avi` and the output extension to `.webm`. The following important attributes should be set:

- **Destination path:** One should set the destination path to either the same directory as the source or to any other directory.
- **Mirror tree:** This option allows one to mirror the source tree of files into another location. It is best to leave this as the default.
- **Copy timestamps:** ensures that the converted file has the same timestamps as the original, this is useful when searching using the dates as this option preserves the original file’s timestamps.

3.7.2 The Tagger processor

Allows the user to run an arbitrary program to set the tags of the media. The tagger program is passed the full path to the media file. It should print out the tag information on standard output. For example, let us say we have a program called `tagger`, when `tagger` is supplied a path, it should print out the following:

```
$ tagger /path/to/image.png
fox:1
temperature:25
completed:True
$
```

Each line should correspond to an existing metadata tag of the project. This output is automatically parsed and the media tags are updated with these values. The tagger processor in the above case should be configured with the command set to `tagger`. The input file argument is automatically passed.

As another example of how one can extract EXIF information and populate specific tags (on OS X or Linux), one may do the following. First make sure `Imagemagick` is installed, specifically the `identify` command is working. Then create a small script, lets call it `extract_exif.sh`:

```
#!/bin/bash
identify -verbose $1 | grep exif | cut -d ':' -f 2-
```

The second line takes the input file, and selects only the exif information. Make sure this script is executable:

```
$ chmod a+x extract_exif.sh
```

When this script is executed as follows, it prints the following:

```
$ extract_exif.sh /path/to/image.jpg
GPSAltitude: 0/1
GPSAltitudeRef: 0
GPSDateStamp: 2012:07:27
GPSInfo: 356
GPSLatitude: 12/1, 56/1, 3333/100
GPSLatitudeRef: N
GPSLongitude: 77/1, 35/1, 5106/100
GPSLongitudeRef: E
[...]
```

One can add a tagger processor and set the command to the full path of this script. Now when one runs the processing, if you have defined any of the tags, for example if a tag called `GPSDateStamp`, `GPSLatitude`, and `GPSLongitude`, then those fields will be extracted and saved in your metadata. This allows you to save only fields that are relevant to your work.

The tagger can be implemented in R or Python for example. In case this is a Python script one may wish to set the command to `python /path/to/script.py`. If `python` is not on the `PATH`, you can also explicitly specify the full path to the Python interpreter. ViXeN will simply pass the file to this script at the end.

The tagger processor can be more complicated than this and do a lot more but this should give you an idea of the power of this approach.

Note: A word of warning when writing R scripts. When R scripts encounter an error, they usually do not exit with an exit status indicating that an error occurred. ViXeN expects that the script exits with a non-zero exit status to determine if the script was successfully run or not. So if you see strange or incorrect results, just check your script for errors.

For EXIF and other metadata information extraction from a variety of media, `exiftool` is a very powerful option that can be used for populating the tags. It is a lot easier to use than the example above and also supports a wide variety of file formats.

3.7.3 The Python processor

This processor allows the user to run arbitrary Python code to set the media tags. This requires knowledge of Python and the ViXeN API but is relatively simple. A simple example is provided below to illustrate the ideas. Let us assume that the following example code is typed into the text box:

```
import os.path
def process(relpath, media, dest):
    media.tags['parent'] = os.path.dirname(media.path)
    media.tags['comment'] = '%s bytes' % media.size
    media.tags['completed'] = True
```

The function is passed three arguments. The `relpath` is the relative path to the media file. The `media` instance is the actual media object associated with the media file. The media object has a `path` attribute, it also has a `tags` attribute which is a dictionary with the keys as the tags for the media. Thus, `media.tags['parent']` is the parent tag. `media.size` is the size of the file in bytes. The above example is a trivial one, one can write arbitrary Python code to process the tags or run external programs if desired. This processor does require reasonable knowledge of Python programming. What it does do is provide a powerful mechanism for scripting the metadata using Python.

CHAPTER 4

Indices and tables

- `genindex`
- `search`