

# CUENCAS Developer's Guide

by

Ricardo Mantilla

2008

## Contents Table

<b><u>CHAPTER 1</u></b>	<b><u>INTRODUCTION</u></b>	<b><u>3</u></b>
1.1	LICENSING	4
1.2	OBTAINING THE SOURCE CODE	4
<b><u>CHAPTER 2</u></b>	<b><u>THE CUENCAS DATABASE STRUCTURE</u></b>	<b><u>6</u></b>
2.1	THE RASTERS FOLDER	6
2.2	THE SITES FOLDER	12
2.3	THE VECTORS FOLDER	14
2.4	THE LINES FOLDER	14
2.5	THE POLYGONS FOLDER	14
2.6	DOCUMENTATION	15
<b><u>CHAPTER 3</u></b>	<b><u>THE HYDROSCALINGAPI</u></b>	<b><u>16</u></b>
3.1	HYDROSCALINGAPI.IO	16
3.2	HYDROSCALINGAPI.MAINGUI	16
3.3	HYDROSCALINGAPI.SUBGUIs	16
3.4	HYDROSCALINGAPI.MODULES	17
3.5	HYDROSCALINGAPI.TOOLS	17
3.6	HYDROSCALINGAPI.UTIL	17
3.7	HYDROSCALINGAPI.EXAMPLES	18
<b><u>CHAPTER 4</u></b>	<b><u>QUICK START GUIDE: PROGRAMMING EXAMPLES USING THE HYDROSCALINGAPI</u></b>	<b><u>19</u></b>
4.1	EXAMPLE 1: IMPLEMENTING NEW HILLSLOPE DYNAMICS FOR THE RAINFALL-RUNOFF MODULE	19
4.2	EXAMPLE 2: IMPLEMENTING A GEOMORPHIC NETWORK ANALYSIS	19
4.3	EXAMPLE 3: CREATING A NEW MODULE TO CREATE INPUTS FOR TRIBS	19

# Chapter 1

## Introduction

CUENCAS is a river networks analysis tool. It can be seen as a simplified version of the original GIS called HidroSig, developed by the Water Resources Department at the Universidad Nacional de Colombia over the last 10 years.

While HidroSig has evolved to be a full-featured GIS, CUENCAS has kept a narrow focus on river networks, and it is mainly intended for three tasks: *i*) river networks extraction and analysis, *ii*) numerical simulation environment of rainfall-runoff processes and *iii*) data archiving framework.

CUENCAS is meant to be a research tool, rather than a commercial product. This is reflected in the GNU-GPL license under which the source code is distributed. The license is discussed in greater detail in section 1.1. The development of CUENCAS since 2001 has been supported by various grants from the National Science Foundation (NSF).

This document is intended for programmers interested in further development of the software. It is assumed throughout the document that the reader is familiar with the Java™ language. It is also recommended that the reader familiarize him/herself with the VisAD graphics library (<http://www.ssec.wisc.edu/~billh/visad.html>). Most of the graphical interfaces use the VisAD library to display data, thus VisAD is deeply embedded everywhere in the code.

This document contains all the information related to the program structure with specific examples that will help a programmer get familiar with the many Java™ objects in the hydroScalingAPI. An online uptodate Javadoc of the hydroScalingAPI package complements this document (<http://cires.colorado.edu/~ricardo/cuencas/javadoc/index.html>).

The JavaDoc provides detailed information about the purpose of the specific methods contained in each class.

In order to develop code to enhance or improve CUENCAS it is necessary to be familiar with the CUENCAS-database structure and with the variety of classes in the hydroScalingAPI package. These two components are explained in detail in chapters Chapter 2 and Chapter 3, respectively. In addition some examples of programming using the hydroScalingAPI are given in Chapter 4. Chapter 4 is also designed for programmers interested in learning the code from the inside using a hands-on approach.

## 1.1 Licensing

CUENCAS is distributed under the GNU-GPL license. The GNU General Public License is intended to guarantee your freedom to share and change free software and to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it.

The exact terms of the license are distributed with the source code. They can also be found at <http://www.gnu.org/licenses/gpl.html#SEC3>.

## 1.2 Obtaining the Source Code

The source code can be obtained using a Subversion client from the repository at <svn://epscor.nmt.edu/src/cuencas/>. The code can be checked out anonymously, however if you are interested in including your changes into the master CUENCAS source code you will need to contact us in order to obtain a username and password that allows you to commit your changes. You can do this by contacting Ricardo Mantilla at [ricardo@cires.colorado.edu](mailto:ricardo@cires.colorado.edu).

The source code structure is organized as a Netbeans 6.0.1 project. Thus it is recommended to use the Subversion module built for Netbeans 6.01 to obtain and manage the source code. Following the steps given here will allow you to successfully download and interact with the source code. Both Netbeans and Subversion are freely available for a wide variety of platforms including Linux, Windows, and Mac OS X.

Follow these steps carefully:

1. **Install the Subversion client:** A subversion client can be obtained at <http://subversion.tigris.org/>. Test installation with `svn -h` in the command prompt.
2. **Install Netbeans 6.0.1 which includes the Subversion module:** Netbeans can be obtained at <http://www.netbeans.org/>
3. **Checkout code:** Go to the Subversion menu and select the “Checkout” option. In the pop-up window type <svn://epscor.nmt.edu/src/>. Write the username and password provided by the CUENCAS Developers Group. In the next pop-up window click the “Browse” button in the Repository Folder and select the “**cuencas**” folder. Finally select the directory where the CUENCAS source code to be stored by clicking the Browse button in the Local Folder section. Hit the Finish button, and click accept when Netbeans prompts you to open the recently downloaded project.
4. **Close the Reference problem window that pops-up.**
5. **Create the VisAD Library:** Right-click on the “cuencas” project (left-side column) and select the “Resolve Reference Problems” item. You will get the message “VisAD library not found”. Click the “resolve” button. In the pop-up

window create a new library called “VisAD” (case sensitive), add the appropriate \*.jar files for each of 3 tabs. The three \*.jar files are distributed along with the source code in the “external” subfolder. Finally close the Reference problem window that appears.

6. **Compile the code and build the Javadoc:** Go to the menu Build → Clean and Build Main Project. Then Go to the menu Build → Generate Javadoc for “cuencas”.
7. **Start Modifying the code:** You are now ready to start making changes to the source code. Don’t forget to keep your code updated (Subversion → Update All)<sup>1</sup>. This will avoid conflicts when you decide to commit your changes (Right click on the file to commit or the project if you modify multiple files and select Subversion → Commit). Add a clear summary of the changes that you perform and the reason behind them.

---

<sup>1</sup> After you update your code it is convenient that you Build → Clean and Build Main Project.

# Chapter 2

## The CUENCAS Database Structure

A CUENCAS Database is essentially a well-organized hierarchy of folders in the computer. This was done to reduce the complexity of accessing and interacting with the data. The hydroScalingAPI is a java set of libraries that allow high-level interaction with the database, however files for the database can be created using programs like Matlab, IDL, etc.

A Folder is considered to be a Cuencas Database if it contains six subdirectories labeled: Documentation, Lines, Polygons, Rasters, Sites, and Vectors, and 2 files: logo.jpg and name.txt.

The first file is a standard JPG image with dimensions 500x50 pixels, with the desired logo for the database, and the second file is a plain text document with the name you want for the database.

Each directory is meant to hold different types of information with specific file formats that will be detailed later. It is expected that all information in a given database have a common integrating idea, which can be a basin location, a region or a country.

Note: The folders structure can be automatically created by CUENCAS by selecting the menu "File → Create a Data Base" and then selecting an empty folder.

In the following sections details are given about the contents of each of the folders mentioned above.

### 2.1 The Rasters Folder

#### 2.1.1 Purpose

This is perhaps the most important folder. This folder contains all the spatially distributed fields. Data in this folder has been classified in two groups (additional subfolders): Topography and Hydrology.

The Topography subfolder stores all the information about Digital Elevation Models (DEMs) and is treated in a special manner by CUENCAS. The folder can contain a single DEM or a collection of DEMs. These can be organized in multiple subdirectories to facilitate its use.

The Hydrology folder is meant to contain every other spatial field, including precipitation, evaporation, land use, vegetation, and temperature fields.

The user can create as many subfolders inside these two folders as needed to classify its own specific information. The data format is proprietary and is described in the following section.

This folder can be populated by means of importing DEMs or hydrometeorological fields from other formats using the tools in the menu ``File → Import → Import DEM" or ``File → Import → Import Hydroclimatic Raster"

## 2.1.2 Formats

### 2.1.2.1 Format for files in the Topography folder (DEMs)

A topography data-file is actually determined by two files with the same name: a metafile (with extension \*.metaDEM) and a binary (XDR) data file (with extension \*.dem). The metafile is an ASCII file that contains all the information necessary to read the data in the binary data file. Currently there are twelve tags,:

1. **[Name]**: A string of ASCII characters with a short name for the variable. We recommend to keep it short, since additional information about the variable can be given under the **[Information]** tag.
2. **[Southernmost Latitude]**: Given in the deg:min:sec format (e.g. 39:33:49.50 N)
3. **[Westernmost Longitude]**: Given in the deg:min:sec format (e.g. 110:35:49.50 W)
4. **[Longitudinal Resolution (ArcSec)]**: A float numeric value given in arc-seconds.
5. **[Latitudinal Resolution (ArcSec)]**: A float numeric value given in arc-seconds.
6. **[# Columns]**: The number of columns for the data array
7. **[# Rows]**: The number of rows for the data array.
8. **[Format]**: A string indicating the data format. It can be either Byte, Integer, Float, or Double
9. **[Missing]**: A value identifying missing data
10. **[Temporal Resolution]**: For DEMs this field is always set to "Fix"
11. **[Units]**: Use the International System (e.g. m for meters, km for kilometers, etc)
12. **[Information]**: All the relevant information about this DEM.

The binary file is written in the External Data Representation Standard (XDR). This is the native format for Java I/O operations. The **[Format]** tag accepts the values: Byte, Integer, Float, and Double.

Once a DEM is processed by CUENCAS (see NetworkExtractionModule in section 3.4.3), a set of files with the same name as the DEM file are created in the folder where the DEM resides. These files are binary files with the same dimensions of the de DEM. They contain a variety of geomorphic information. The format and contents of these files are:

- \*.areas            A file of Floats. Each position contains the upstream area (in Km<sup>2</sup>) of that location in the DEM.
- \*.corrDEM        A file of Doubles. Each position contains the modified DEM pixel. Pixels are modified in such a way that every position in the matrix contains one and only one lowest nearest neighbor.
- \*.dir             A file of Bytes. Each position contains a number between 1 and 9 indicating the "flow" direction. 1-SW, 2-S, 3-SE, 4-W, 6-E, 7-NW, 8-N, 9-NE.
- \*.dtopo          A file of Integers with the same dimension of the DEM containing the Maximum Upstream Topologic Distance.
- \*.gdo            A file of Float. Each position contains the distance (in Km) to the border of the DEM, the ocean or an internal lake.
- \*.horton         A file of Bytes. Each position contains the Horton-Strahler order of the stream in which the pixel is contained.
- \*.lcp            A file of Floats. Each position contains the longest channel length (in Km) up to the pixel.
- \*.ltc            A file of Floats. Each position contains the total channel length (in Km) of the river network upstream of the pixel
- \*.magn          A file of Integer. Each position contains the Magnitude (according to \cite{shre1966})
- \*.mcd            A file of Floats. Each position contains the Maximum Channel Drop (in m) along the streams.
- \*.redRas         A file of Bytes. Each position contains either 0 or 1 indicating the pixels that contain a channel.
- \*.slope         A file of Doubles. Each position contains the slope in the direction of the maximum gradient.
- \*.tcd            A file of Floats. Each position contains the total drop (in m) along the channels.
- \*.tdo            A file of Integers. Each position contains the Topologic Distance to the border of the DEM, the ocean or an internal lake.

In addition, some auxiliary files are created. These are:

- \*.ap            Area-Slope of network points. Usefull for the area-slope analysis for network pruning.
- \*.log          An ASCII file containing a record of the outlets that have been selected by the user using the GUI.

Finally, the three files that describe the network topology are: `*.stream`, `*.point`, and `*.link`.

These files have been developed as an appropriate way to represent a river network based on the hierarchical ordering structure proposed by Horton (1945), and later modified by Strahler (1964), known today as the Horton-Strahler ordering scheme. Horton-Strahler order has been proven to be an ideal parameter to study scale-invariant or statistical scaling properties of morphometric quantities in river networks. The so-called Horton laws of drainage composition introduced by Horton (1945) were the very first representation of scaling in the means of complete order streams. Peckam et al. (1999) generalized Horton laws from means to rescaling of probability distributions of complete Horton-Strahler streams in real networks, and called it distributional simple scaling. Veitzer (2000) introduced a new class of random self-similar networks (RSNs) that were shown to exhibit distributional simple scaling in drainage areas and stream lengths.

The `*.stream` file assigns six properties to each complete Horton-Strahler stream in a network:

- **Head-id:** The pixel-ID<sup>2</sup>
- **Contact-id:** The pixel-ID immediately upstream of the junction where the stream ends.
- **Tail-id:** The pixel-ID of the junction where the streams ends.
- **Order:** The Horton-Strahler order for this stream.
- **Ini-Link:** The position of the first link in the stream in the `*.link` file.
- **Num-Links:** The number of links in this stream.

The `*.link` file assigns six properties to each link in the network:

- **Magnitude:** The link magnitude as defined by \cite{shre1967}.
- **Head-id:** The pixel-ID where the links starts.
- **Contact-id:** The pixel-ID immediately upstream of the junction where the link ends.
- **Tail-id:** The pixel-ID of the junction where the link ends.
- **Ini-point:** The position of the first pixel in the link in the `*.point` file.
- **Num-points:** The number of points in this link.

Finally the `*.point` file contains the \$pixel-ID\$s of all the pixels that contain a channel, sorted according to the structure described above.

---

<sup>2</sup> The *pixel-ID* is calculated from the  $x,y$  position in the matrix as  $pixel-ID=y*NC+x$ , where  $NC$  is the number of columns in the matrix. Notice that  $x$  and  $y$  can be easily back calculated from the *pixel-ID* as,  $x=pixel-ID \bmod NC$  and  $y=\mathbf{integer}(pixel-ID/NC)$ , where **mod** represents the operation module, and **integer** represents the integer part of the ratio. where the streams starts.

### 2.1.2.2 Format for files in the Hydrology folder

A hydrology data file is actually a group of two or more files with the same name: there will always be a metafile (with extension \*.metaVHC) and a group of binary (XDR) data files with extension \*. [hhmmss] . [dd] . [month] . [yyyy] . vhc .

The metafile is identical to the one described in the previous section. The group of companion binary files must have the same format, number of columns and rows, and the same spatial coverage (Southernmost latitude and Westernmost longitude). The extension holds the key to understand the time variability of the data, and it has to be consistent with the tag [**Temporal Resolution**] in the metafile. The [**Temporal Resolution**] tag must be followed by the time interval of the data (e.g. 1-minute, 1-second, 1-hour, 1-month, 1-year)

The extension of a vhc file can be manipulated to describe almost any kind of time variability on the data. It will be illustrated by examples to avoid long explanations.

#### Annual Average

In this case there will only be a single companion file for the metafile. The file extension should look like this: **name.average.vhc**

#### Annual Data

In this case there can be as many companion files as necessary. The file extension should look like this: **name.yyyy.vhc**. As an example, for annual accumulated precipitation values, the group of files would look like this:

```
precip.1986.vhc
precip.1987.vhc
precip.1988.vhc
precip.1989.vhc
precip.1990.vhc
precip.1991.vhc
precip.1992.vhc
```

#### Monthly Averages

In this case there will be 12 companion files for the metafile. The file extension should look like this: **name.month.average.vhc**. As an example assume you have evaporation monthly means, the group of files would look like this:

```
evap.January.average.vhc
evap.February.average.vhc
evap.March.average.vhc
.
.
.
evap.October.average.vhc
evap.November.average.vhc
evap.December.average.vhc
```

#### Monthly Data

In this case there can be as many companion files as necessary. The file extension should look like this: **name.month.yyyy.vhc**. As an example, for monthly temperature values, the group of files would look like this:

```
temp.January.2002.vhc
temp.February.2002.vhc
temp.March.2002.vhc
.
.
.
temp.October.2002.vhc
temp.November.2002.vhc
temp.December.2002.vhc
temp.January.2003.vhc
temp.February.2003.vhc
temp.March.2003.vhc
.
.
.
temp.Octover.2004.vhc
temp.November.2004.vhc
temp.December.2004.vhc
```

### Daily Averages

In this case there will be 365 or 366 companion files for the metafile. The file extension should look like this: `name.dd.month.average.vhc`. As an example, for daily mean radiation, the group of files would look like this:

```
radiation.01.January.average.vhc
radiation.02.January.average.vhc
radiation.03.January.average.vhc
.
.
.
radiation.01.February.average.vhc
radiation.02.February.average.vhc
.
.
.
radiation.30.December.average.vhc
radiation.31.December.average.vhc
```

### Daily Data

In this case there can be as many companion files as necessary. The file extension should look like this: `name.dd.month.yyyy.vhc`. As an example, for daily wind speed values, the group of files would look like this:

```
radiation.01.January.1999.vhc
radiation.02.January.1999.vhc
radiation.03.January.1999.vhc
.
.
.
radiation.01.February.1999.vhc
radiation.02.February.1999.vhc
.
.
.
radiation.30.December.1999.vhc
```

```

radiation.31.December.1999.vhc
.
.
radiation.01.January.2000.vhc
radiation.02.January.2000.vhc
radiation.03.January.2000.vhc
.
.
radiation.01.February.2000.vhc
radiation.02.February.2000.vhc
.
.
radiation.30.December.2000.vhc
radiation.31.December.2000.vhc

```

### Hourly, per Minute or per Second Data

In this case there can be as many companion files as necessary. The file extension should look like this: `name.hhmmss.month.yyyy.vhc`. As an example, for precipitation values every 5 minutes, the group of files would look like this:

```

prec.044000.10.September.1964.vhc
prec.044500.10.September.1964.vhc
prec.045000.10.September.1964.vhc
prec.045500.10.September.1964.vhc
.
.
prec.211500.09.September.1964.vhc
prec.222500.09.September.1964.vhc
prec.223000.09.September.1964.vhc
prec.223500.09.September.1964.vhc
prec.224000.09.September.1964.vhc

```

Notice how there is no need of a data file for each and every moment of time. Applications in CUENCAS, such as the rainfall-runoff module, will assume every non existent file as either 0's or missing data.

## 2.2 The Sites Folder

### 2.2.1 Purpose

This folder is designed to contain information about locations and gauges in two subfolders named Locations and Gauges. It is important to those interested in time series analysis, and it is very useful to create reference points that can be later displayed on top of maps. An interesting feature is that images can be associated to the locations created.

### 2.2.2 Formats

Site data has been divided into 2 groups (subfolders) Locations and Gauges. The reason is that Gauges have additional tags that describe the time series contained in the file. Files are in ASCII format, however the files are compressed using the LZW

algorithm. Note that text editors such as VI can edit these files directly. In addition Java includes libraries to read and write compressed files. The specific tags for each type of file is described in the following sections.

Note that the files can be grouped in directories for different types of time series (e.g streamflow, precipitation, temperature, etc) however this is not required.

### 2.2.2.1 Locations

A “Location” is described in an ASCII file by the information associated to tags. Currently there are ten tags, which are detailed detail here:

1. **[type]**: An indicator such as City, GPS Location, Survey point, etc.
2. **[source]**: The source for the data (e.g. USGS).
3. **[site name]**: The name of the location.
4. **[county]**: County.
5. **[state]**: State.
6. **[latitude (deg:min:sec)]**: Given in the deg:min:sec format (e.g. 39:33:49.50 N)
7. **[longitude (deg:min:sec)]**: Given in the deg:min:sec format (e.g. 110:33:49.50 W)
8. **[altitude ASL (m)]**: Altitude above sea level.
9. **[images]**: A list of images associated to the location. If the image is in a file the path is given relative to the location of the Location file. The image can also be a URL. After the path, followed by a semicon there must be a description of the image.
10. **[information]**: Any additional information about the site.

### 2.2.2.2 Gauges

A “Gauge” is described in an ASCII file by the information associated to tags. Currently there are fifteen tags, which are detailed detail here:

1. **[code]**: A string of ascii characters with the gauge code. Codes are typically assigned by the source agency.
2. **[agency]**: The source agency (e.g. NOAA, USGS, etc)
3. **[type]**: The type of data contained in the file (e.g. Streamflow, Precipitation, Temperature).
4. **[site name]**: A name for the gauge.
5. **[stream name]**: This tag is mainly for streamflow locations.
6. **[county]**: County
7. **[state]**: State

8. **[data source]:** A brief description of how the data was obtained.
9. **[latitude (deg:min:sec)]:** Given in the deg:min:sec format (e.g. 39:33:49.50 N)
10. **[longitude (deg:min:sec)]:** Given in the deg:min:sec format (e.g. 110:33:49.50 W)
11. **[altitude ASL (m)]:** Altitude above sea level.
12. **[drainage area (km<sup>2</sup>)]:** Upstream accumulated area
13. **[data units]:** use the international units system
14. **[data accuracy]:** use the international units system
15. **[data (yyyy.mm.dd.hh.mm.ss value)]:** The actual data.

## 2.3 The Vectors Folder

### 2.3.1 Purpose

This folder hold vector information, such as roads, blue lines, etc. The folder accepts two kinds of formats: Shapefiles and DLGs. More information about these two data formats is provided here.

### 2.3.2 Formats

CUENCAS can read two types of data format. DLGs from the USGS and ESRI Shapefiles.

## 2.4 The Lines Folder

### 2.4.1 Purpose

Similar to vectors, but in this case, additional properties can be assigned to the lines, for added functionality (not implemented yet)

## 2.5 The Polygons Folder

### 2.5.1 Purpose

Similar to vectors, but in this case lines are expected to form a closed polygon. Polygons are used by CUENCAS to integrate a spatially distributed variable over the enclosed region the polygon defines. The obvious application is water balances, where the basin divide defines the polygon and long term mean precipitation and evaporation fields are integrated to get an estimated mean discharge.

### 2.5.2 Formats

Polygon files are intended to represent regions enclosed by a polygon line. The basin divide is the typical example for this kind of data. The file format is ASCII, so they can

be read and modified by any standard text editor (If you are using windows, use wordpad instead of the notepad).

1. **[name]:** A string of ASCII characters it can be any name for the variable. We recommend to keep it short, since additional information about the variable can be given under the **[Information]** tag.
2. **[coordinates lon/lat (deg)]:** The polygon vertices coordinates in decimal degrees. Western longitudes and southern latitudes need to be negative.
3. **[information]:** All the relevant information about the polygon file.

## **2.6 Documentation**

### **2.6.1 Purpose**

This is the most flexible directory in terms of data format requirements. Any file that can be handled by the system (PDF, JPGs, TXT, DOC) can be thrown in here. The idea is to create a depository for all the documentation in the form of papers, reports, images, related to the region being treated, but that cannot be directly linked in an spatial context to the rest of the information in the database.

# Chapter 3

## The hydroScalingAPI

The hydroScalingAPI is a set of classes and interfaces that communicate with the CUENCAS-database, which was described in the previous section. The different classes that compose the hydroScalingAPI have been arranged in sub-packages. We describe the purpose of the classes in each of the sub-packages in the following sections. The specific classes and methods are documented online at <http://cires.colorado.edu/~ricardo/cuencas/javadoc/> using the JavaDoc mechanism. Thus, this document is intended as a conceptual guide to the organization of the API rather than a programmer's guide of the individual classes.

### 3.1 hydroScalingAPI.io

The io package contains all the classes that communicate directly with the files in the database. This is one of the most important set of classes in the hydroScalingAPI. The importance on grouping all these classes is that future modifications of the database structure (i.e. adding new data types, changing database formats, adding/removing tags from metafiles, should not impact any program beyond this package. In consequence, all the classes that require data from the database should access it through the methods provided by the classes in this package. Note that writing directly into the database can be done from different sources.

### 3.2 hydroScalingAPI.mainGUI

This package contains the GIS interface of CUENCAS. This package contains all the data managers for different types of data in the CUENCAS database. It also contains small widgets (GUIs) used by the main GIS interface to provide basic information to the user. The class ParentGUI displays the CUENCAS main interface shown in Fig. XX

### 3.3 hydroScalingAPI.subGUIs

Support classes for GUIs embedded inside the main CUENCAS GUI hydroScalingAPI.mainGUI. It also contains the GUIs embedded inside the main CUENCAS GUI.

### **3.4 hydroScalingAPI.modules**

Contains all the individual modules (advanced tools) of River Network oriented analysis.

#### **3.4.1 hydroScalingAPI.modules.analysis\_TS**

GUIs related to the Time Series Analysis module.

#### **3.4.2 hydroScalingAPI.modules.networkAnalysis**

Classes designed for advanced geomorphic analysis of the River Network structure, and GUIs for to the Network Analysis module.

#### **3.4.3 hydroScalingAPI.modules.networkExtraction**

\label{modules.networkExtraction}

Classes on this package perform DEM analysis to extract the direction matrix and derive a river network structure, and GUIs for the Network Extraction Module.

#### **3.4.4 hydroScalingAPI.modules.rainfallRunoffModel**

Classes that establish the kernel of link-hillslope based rainfall-runoff simulations, and A GUI to the link-hillslope based rainfall-runoff package.

#### **3.4.5 hydroScalingAPI.modules.rsnFlowSymulations**

Classes for rainfall-runoff simulations on Random Self-similar Networks (RSNs).

#### **3.4.6 hydroScalingAPI.modules.tRIBS\_io**

Classes that establish the kernel of link-hillslope based rainfall-runoff simulations, and A GUI to the link-hillslope based rainfall-runoff package.

### **3.5 hydroScalingAPI.tools**

Various generic tools that are used by other objects in the hydroScalingAPI.

### **3.6 hydroScalingAPI.util**

It contains ...

#### **3.6.1 hydroScalingAPI.util.database**

A database engine to manage site-type data.

#### **3.6.2 hydroScalingAPI.util.fileUtilities**

Some classes to handle file filtering and sorting.

### **3.6.3 hydroScalingAPI.util.geomorphology**

Classes that represent the basic geomorphic decomposition units (Basin, hillslope, streams and links).

### **3.6.4 hydroScalingAPI.util.ordDiffEqSolver**

Provides classes and interfaces to solve coupled non-linear ordinary differential equations.

### **3.6.5 hydroScalingAPI.util.plot**

Classes for basic XY plots.

### **3.6.6 hydroScalingAPI.util.probability**

Provides classes and interfaces to create and handle random variables.

### **3.6.7 hydroScalingAPI.util.randomSelfSimilarNetworks**

Classes to create Random Self-similar Networks (RSNs) using recursive replacement algorithm.

## **3.7 hydroScalingAPI.examples**

This package contains examples on how to 1) create spatial fields for the CUENCAS database, 2) use the GeoTransform package to convert degrees to UTM coordinates and vice versa, 3) use CUENCAS objects to obtain relevant geomorphic and topologic properties from specific basins, 4) test capabilities to read and write DLG files, 5) import miscellaneous data into the CUENCAS database, and 6) setting up Rainfall-Runoff simulations for specific purposes without using the GUI.

# **Chapter 4**

## **Quick Start Guide: Programming Examples using the hydroScalingAPI**

- 4.1 Example 1: Implementing New Hillslope Dynamics for the Rainfall-Runoff Module**
- 4.2 Example 2: Implementing a Geomorphic Network Analysis**
- 4.3 Example 3: Creating a new Module to Create Inputs for tRIBS**