

ATExplorer 1.0 - User Manual

Smith Lab, Allen Institute for Brain Science

October 2018

Preface

AT introduction here..

The ATExplorer application integrates a workflow, as well as a number of software components that are useful in the context of Array Tomography .

The following software components are the main building blocks that ATExplorer is built on top of:

- **RENDER PYTHON** by F Collman et al. RenderPython is a thin Python wrapper for *Render*.
- **RENDER** by ??? et.al
- **FIJI** by et. al....
- **DOCKER** et. al.

In addition to the above, semi specialized software packages, a number of open source, C++, libraries are employed by the ATExplorer application:

- **VTK** by
- **POCO** by ??? et.al
- **LIBCURL** by et. al....
- **TINYXML2** et. al.
- **DUNE SCIENTIFIC LIBRARY (DSL)** et. al.

The ATExplorer application was designed and implemented in the lab. of Stephen J Smith and

Forrest Collman, at the Allen Institute of Brain Science by Totte Karlsson.

The following people has been contributing to the effort;

Part One

1	Overview of the ATE Explorer UI and API .	7
1.1	Introduction	
1.2	An Array Tomography workflow	
1.3	The Render Service	
1.4	The ATE Explorer UI	
1.5	Python Bindings	
	Appendices	11
A	Setup a RenderHost and Python	15
B	Python Enabled API's	17
C	Software Design and Software Components	19
C.1	ATE Explorer UI	
C.2	ATE Explorer Software API's	
C.3	ThirdParty libraries	
D	How to Build ATE Explorer and its API's ..	21



1. Overview of the ATEplorer UI and API

1.1 Introduction

This chapter gives an overview of the software that is named *ATEplorer*.

The ATEplorer application was designed and implemented due to an emerging need to allow *non-programmers* to process, manage and explore Array Tomography data on a routine basis.

Depending on the actual protocols, an Array Tomography data set can range in size from a few hundred megabytes, to very large, like several Terra bytes. Depending on the number of stains and sessions, the complexity of the data-sets range from trivial to complex.

One of the main challenges in Array Tomography is the precise digital reconstruction of an original volume, i.e. from individually cut and imaged slices of tissue.

However, before volume reconstruction can begin, various pre data processing algorithms need to be applied, such as median filtering, flat-field correction and de-convolution.

These processing algorithms are all, to some extent, complex. ATEplorer provides the non-programmer user with intuitive and easy to use UI components to guide through this process, in order to quickly get to data that is useful for scientific discovery and exploration.

1.2 An Array Tomography workflow

- MEDIAN CALCULATION

- FLATFIELD CORRECTION
- DECONVOLUTION
- STITCHING
- REGISTRATION
- ROUGH ALIGNING
- FINE ALIGNING
- OTHER

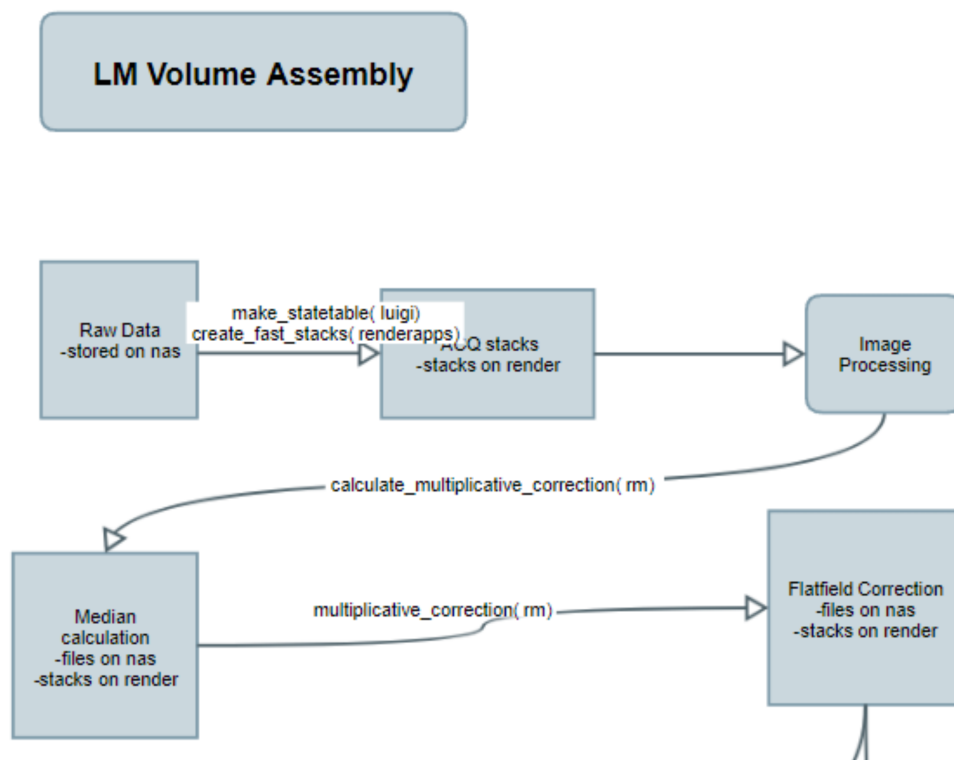


Figure 1.1: Processing

1.3 The Render Service

1.4 The ATEplorer UI

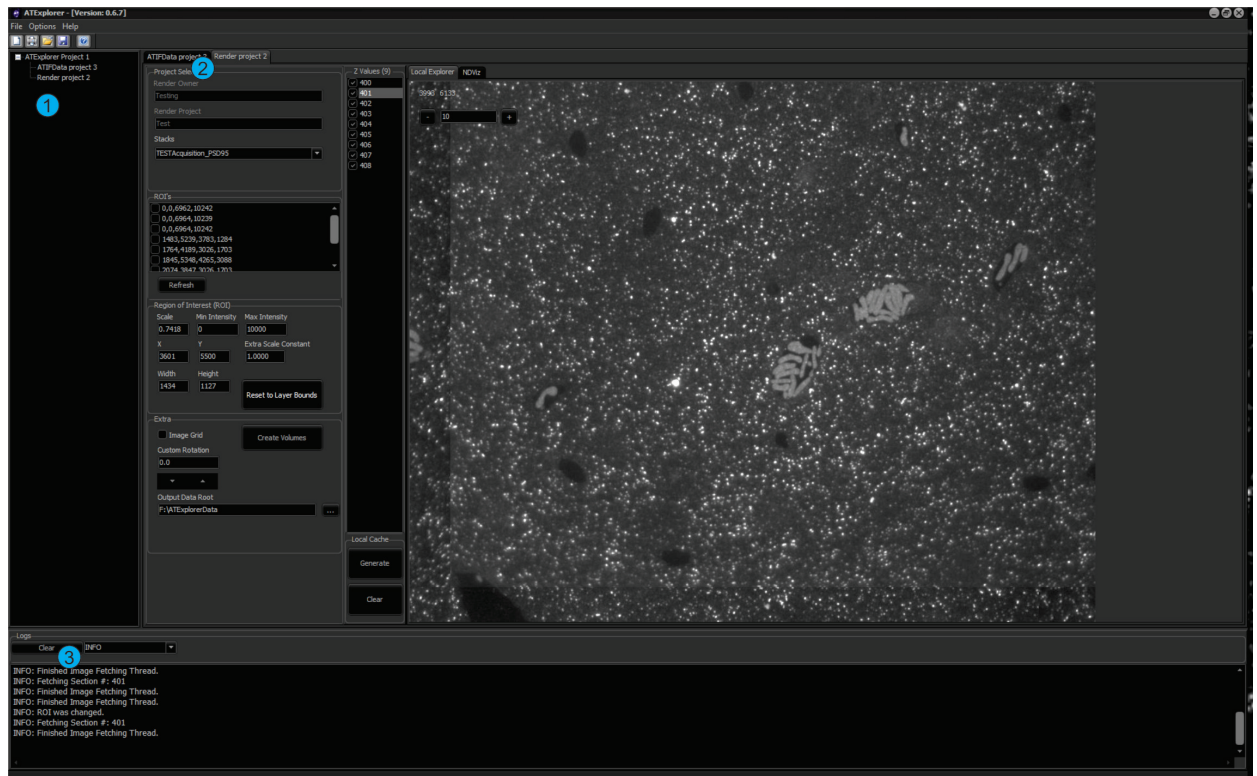


Figure 1.2: ATEplorer UI. The circled numbers in the figure indicate relevant elements of the UI; ① Project(s) TreeView. ② Tabbed Project Item View. ③ Information and Application Log Messages.

1.4.1 Importing Data

- **IMPORTING PROCESS** Give an overview on what happens when data is being imported to ATEplorer .
- **DATA FORMATS** Describe the Allen Institute format, and Kristinas format.

1.4.2 Connecting to a Remote (or local) RenderHost

1.4.3 Creation of RenderStacks

1.4.4 Managing Stacks in Render

1.4.5 Exploring Data

1.5 Python Bindings

Appendices

A	Setup a RenderHost and Python	15
B	Python Enabled API's	17
C	Software Design and Software Components	19
C.1	ATExplorer UI	
C.2	ATExplorer Software API's	
C.3	ThirdParty libraries	
D	How to Build ATExplorer and its API's	21



A. Setup a RenderHost and Python

AT Deployable

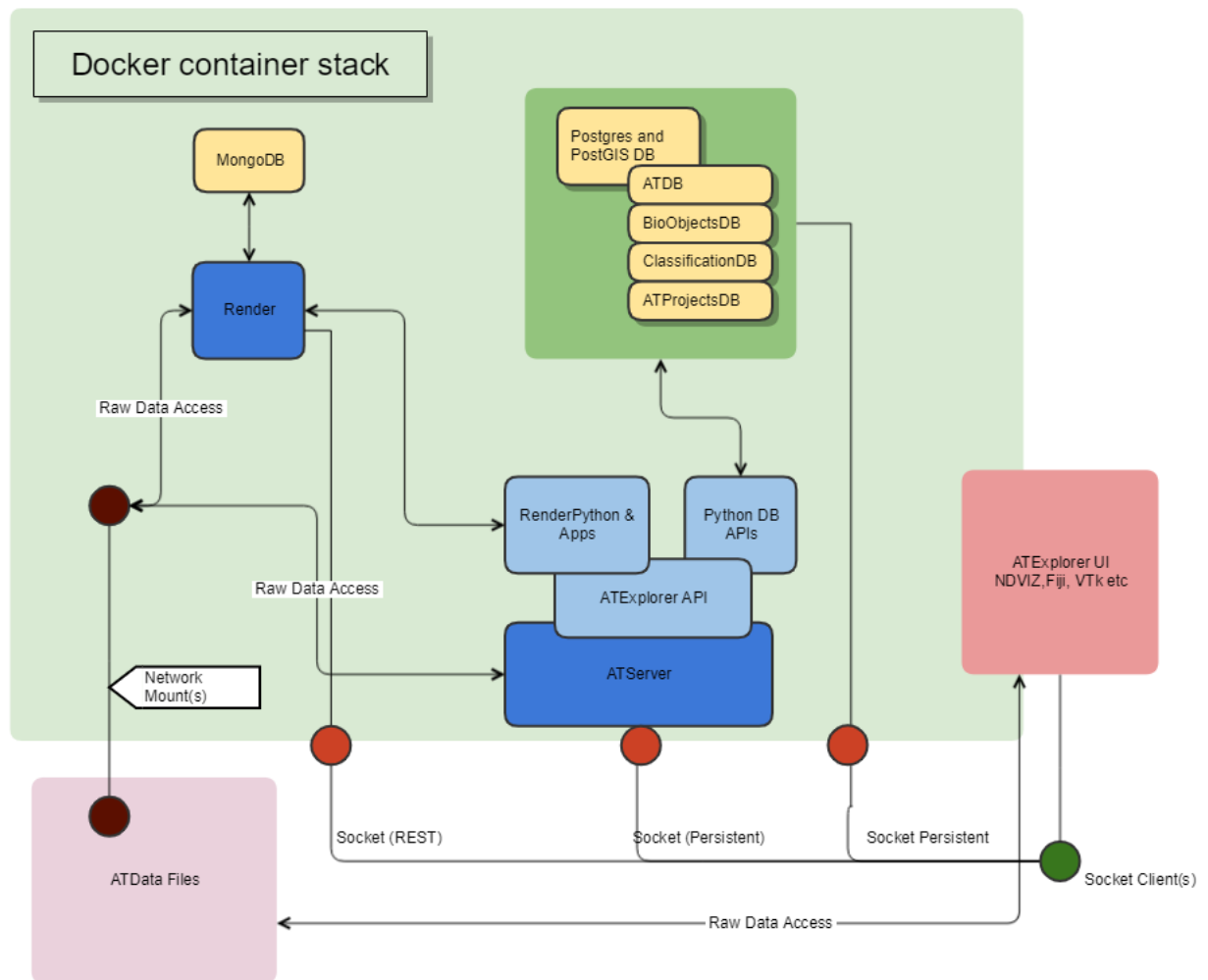


Figure A.1: A deployed system



B. Python Enabled API's



C. Software Design and Software Components

C.1 ATEplorer UI

The ATEplorer UI is Microsoft Windows desktop application implemented using Embarcadero's C++ Builder tools. This environment provide a programmer with hundreds of components for efficient and rapid development of Windows applications.

In addition, thousands of third party components are available as well. This appendix discusses some of the software designs used when implementing the ATEplorer

C.1.1 Observers and subjects

The Tree view and PageControl .

C.1.2 The TreeView

The items in the Treeview stores data objects as (void*) pointers. Any object registered with the tree (as a node) need to be a descendant of the class ExplorerObject. Typical scenario:

```
ExplorerObject* eo = (ExplorerObject*) node->Data
```

```
if(dynamic_cast<...>(eo))
```

```
{
```

```
...
```

}

C.1.3 Populating an ATData object

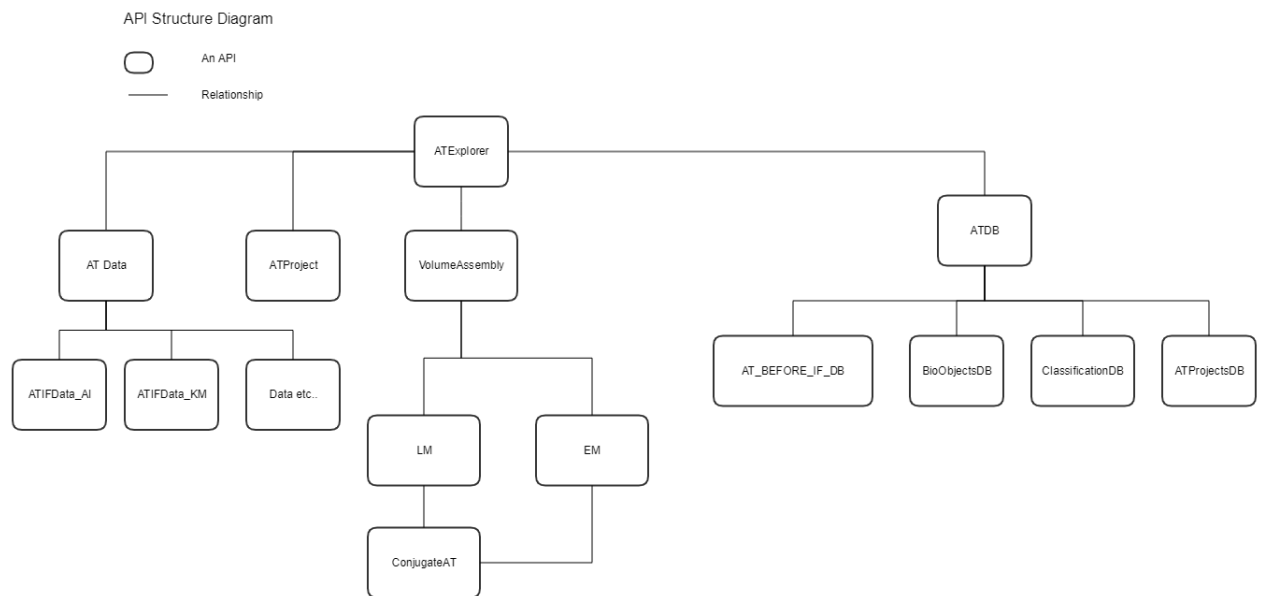


Figure C.1: An overview of some of ATE Explorer API's

C.2 ATE Explorer Software API's

C.2.1 atCore

C.2.2 atData

C.2.3 atVCLCore

C.3 ThirdParty libraries

C.3.1 Poco

C.3.2 libCurl

C.3.3 SQLite 3

C.3.4 TinyXML2

C.3.5 Dune Scientific libraries: dslFoundation



D. How to Build ATEplorer and its API's

Public Software Repository: [git@github.com : TotteKarlsson/ATEplorer.git](https://github.com/TotteKarlsson/ATEplorer.git)