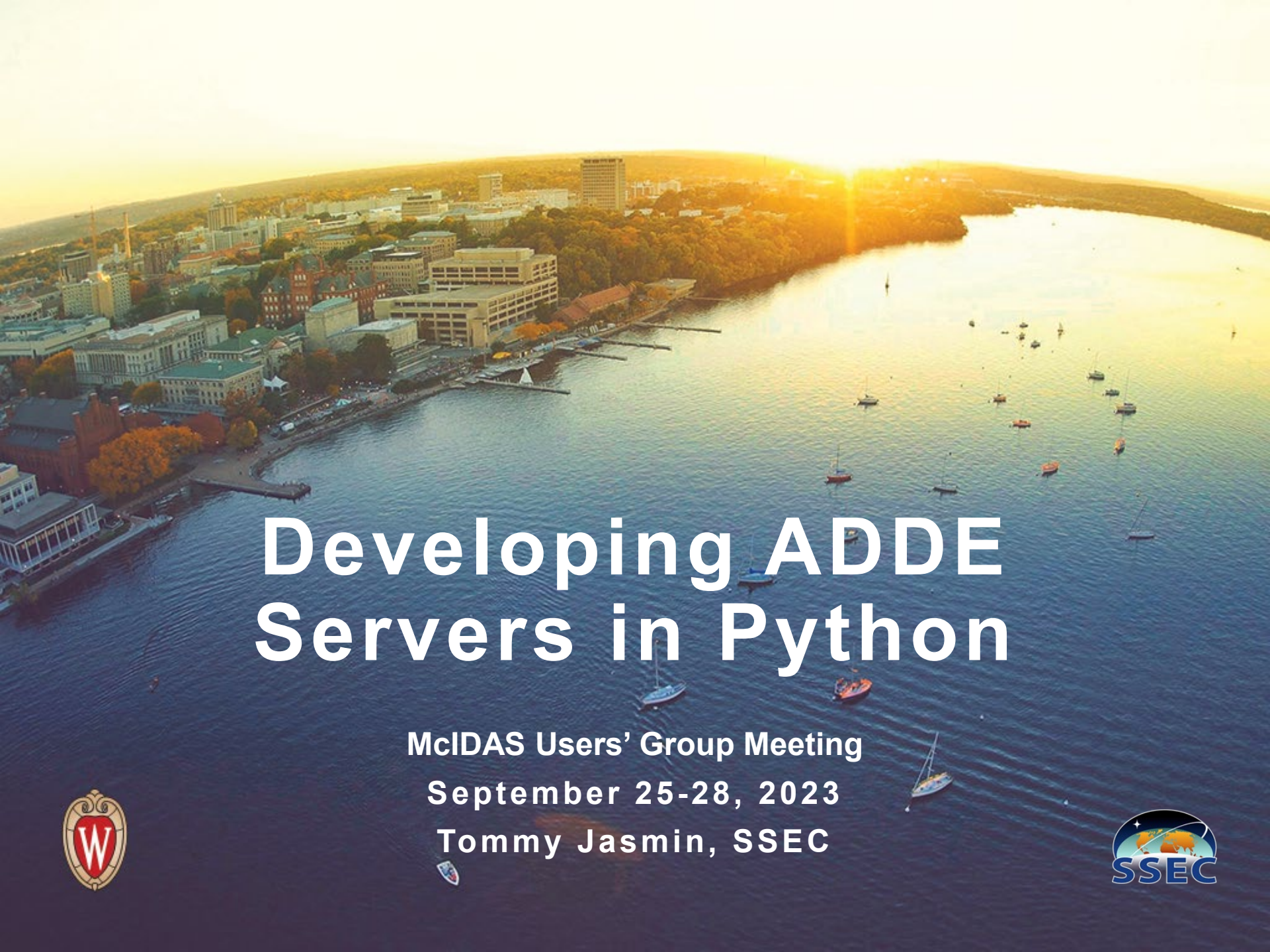


Developing ADDE Servers in Python

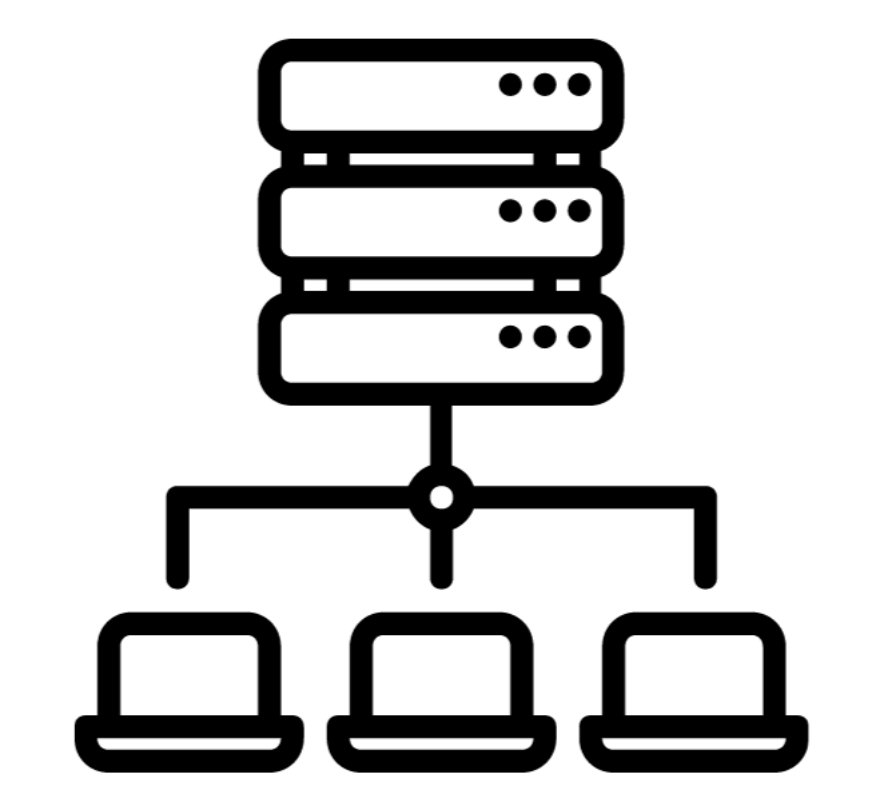
McIDAS Users' Group Meeting

September 25-28, 2023

Tommy Jasmin, SSEC

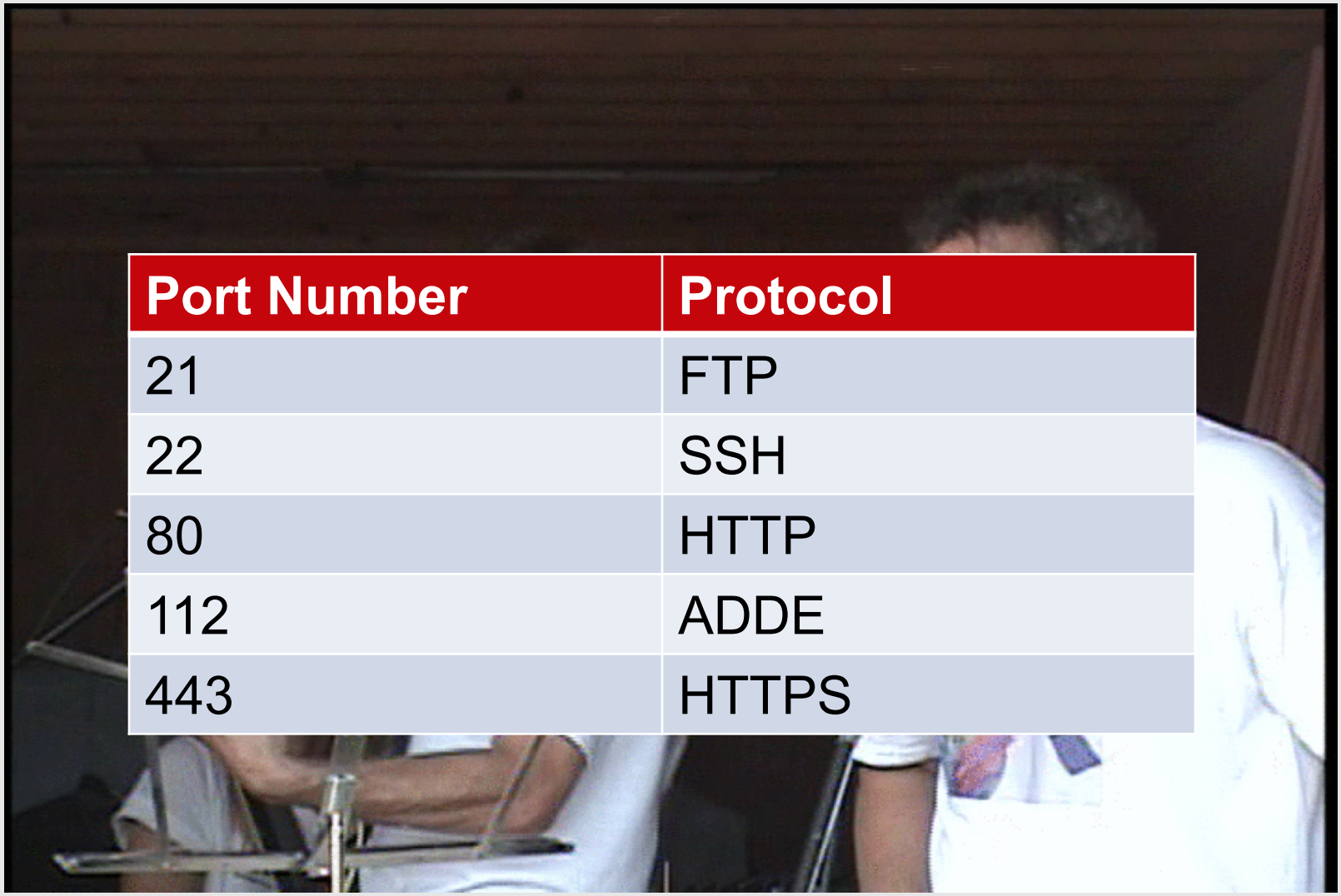
Abstract Data Distribution Environment

Efficient access to large datasets, stored in a variety of file formats, delivered in a common format



- **Standard Client/Server Paradigm**
- **Some Clients: IDL, McIDAS, Matlab, IDV**
- **Servers can be Local, or Remote over TCP/IP**

ADDE Down Through the Ages



Port Number	Protocol
21	FTP
22	SSH
80	HTTP
112	ADDE
443	HTTPS

Under the Hood, an Elegant Simplicity?

A Specialized URL is Sent from Client to Server

adde://server.domain/request?key1=value1&key2=value2…keyN=valueN

Each Key/Value Pair Further Defines and Refines the Request

Similar to a Dynamic Web Page Request, e.g.:

https://mcidas.ssec.wisc.edu/inquiry-v/?inquiry=2499

Example of a Real Request:

adde://easta/imagedata?group=ABI&descr=FD&pos=0&band=2&mag=-8

PyADDE Motivation

Then: Old guys writing servers in Fortran and C.

Now: Young kids writing code quickly in Python

We want to provide a modern, modular, and extensible development framework for current and future developers.

We want to leverage very powerful, strongly supported Python libraries (like Satpy) to abstract away much of the heavy lifting.

Question: Is ADDE still relevant?

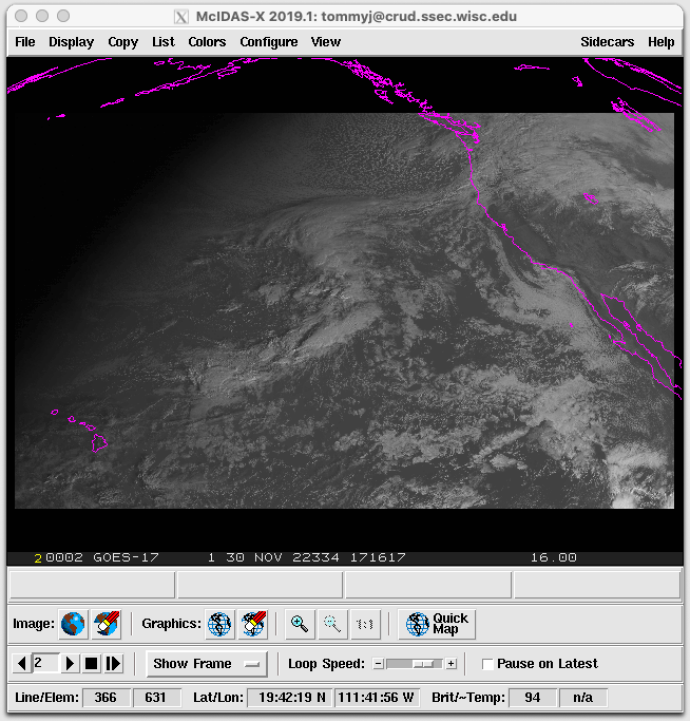
Accomplishments

- Developed a prototype, verified client/server communication needs
- Developed a server which is a functional equivalent of an existing server (GOES-R ABI)*
- Began refactoring, with an API in mind
- Addressed bugs and discrepancies with core
- Developed new servers for new datasets (GK-2A AMI, MTG FCI, still in progress)

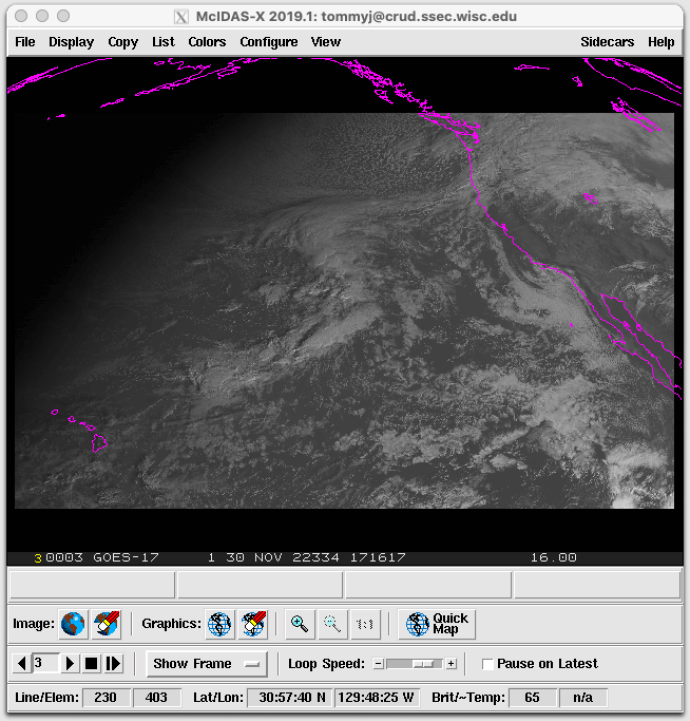
Still to do:

- Another round of refactoring
- Speed them up
- Make some pudding

Why Start with an Existing Server?



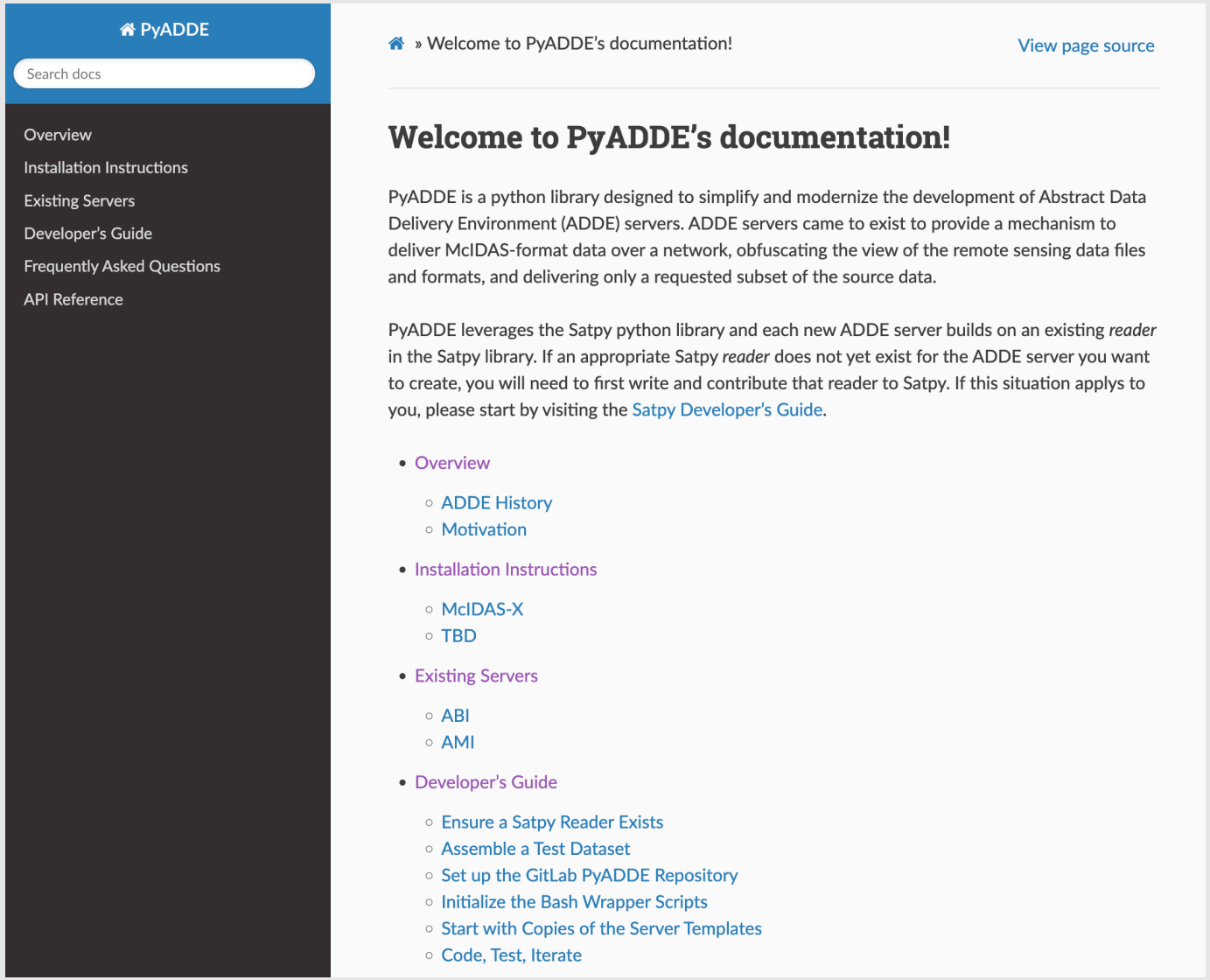
FORTRAN and C



Python

```
L: IMGDISP ABDB17/CONUS.208 LAT=30 137 BAND=1 DAY=22334 MAG=-8
R: IMGDISP ABIP17/CONUS.208 LAT=30 137 BAND=1 DAY=22334 MAG=-8
```

Online Documentation



PyADDE

Search docs

Overview
Installation Instructions
Existing Servers
Developer's Guide
Frequently Asked Questions
API Reference

» Welcome to PyADDE's documentation! View page source

Welcome to PyADDE's documentation!

PyADDE is a python library designed to simplify and modernize the development of Abstract Data Delivery Environment (ADDE) servers. ADDE servers came to exist to provide a mechanism to deliver McIDAS-format data over a network, obfuscating the view of the remote sensing data files and formats, and delivering only a requested subset of the source data.

PyADDE leverages the Satpy python library and each new ADDE server builds on an existing *reader* in the Satpy library. If an appropriate Satpy *reader* does not yet exist for the ADDE server you want to create, you will need to first write and contribute that reader to Satpy. If this situation applys to you, please start by visiting the Satpy Developer's Guide.

- Overview
  - ADDE History
  - Motivation
- Installation Instructions
  - McIDAS-X
  - TBD
- Existing Servers
  - ABI
  - AMI
- Developer's Guide
  - Ensure a Satpy Reader Exists
  - Assemble a Test Dataset
  - Set up the GitLab PyADDE Repository
  - Initialize the Bash Wrapper Scripts
  - Start with Copies of the Server Templates
  - Code, Test, Iterate

Sphinx Python Documentation Generator
https://www.sphinx-doc.org/en/master/

Auto-Generated Developer's API

`pyadde.abi_util`

Collection of Python ABI ADDE utility functions

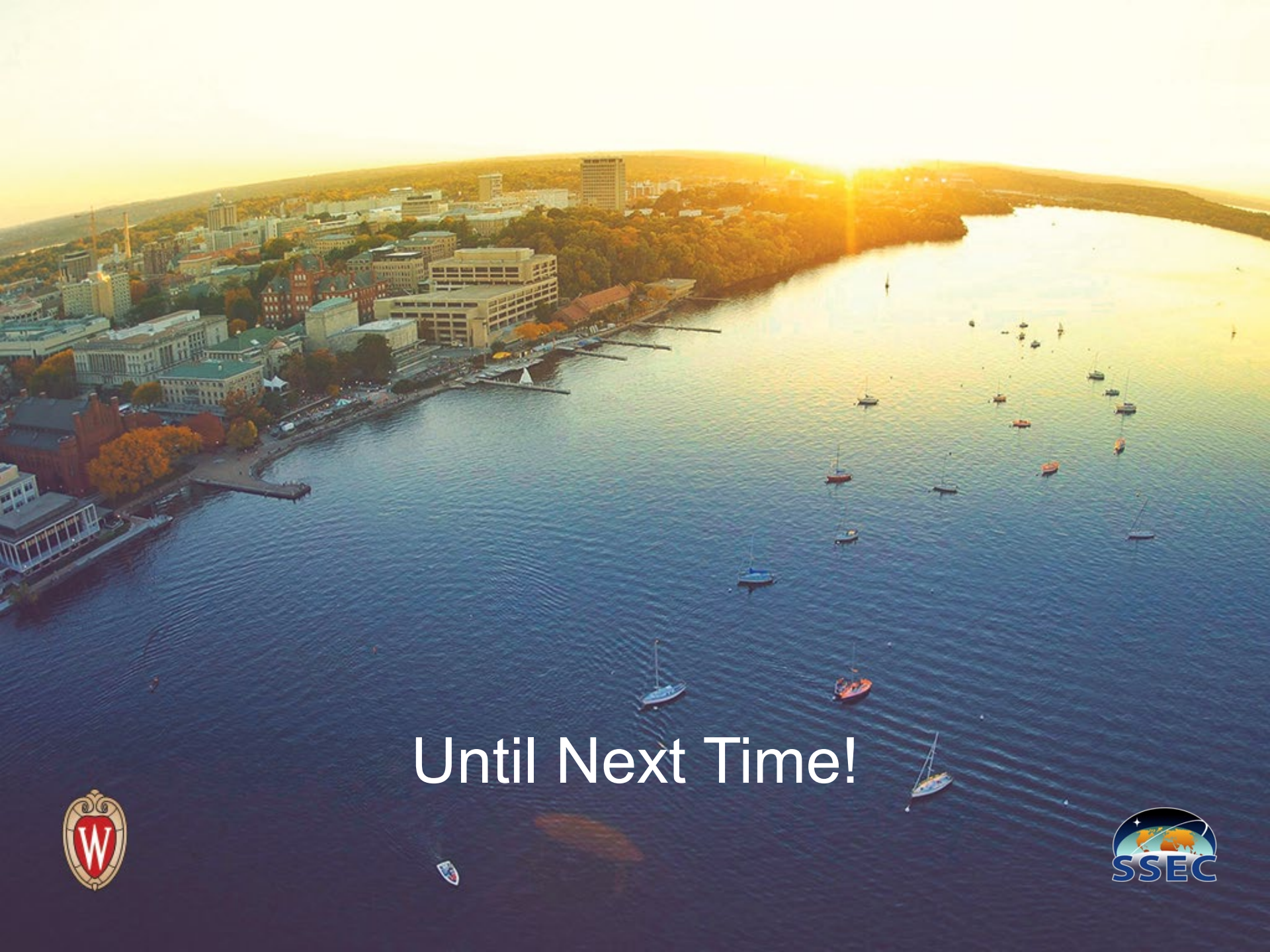
Module Contents

Classes

`ABICalibration`	Generic enumeration.
`ABICalBand`	Structure with big endian byte order
`ABICalFill`	Structure with big endian byte order
`ABICalBlock`	Structure with big endian byte order
`ABICommentBlock`	Structure with big endian byte order

Functions

`coarsest_res`	(channel_list) → str
`abi2bandnum`	(abi_path: str) → str
`is_visible_band`	(band_num: int) → bool
`find_latlon_resolution`	(area: pyresample.geometry.AreaDefinition) → Tuple[float, float]
`alb_to_grayscale`	(data_vals, requested_brit_stretch, band_num)
`temp_to_grayscale`	(data_vals, requested_brit_stretch, band_num)
`lltora`	(rrlat: float, rrlon: float, pplat: float, pplon: float) → Tuple[float, float]
`format_dqf`	(readers, channel: str, attr: str) → str
`create_comments`	(scene: satpy.Scene, central_lat: float, central_lon: float, resolution: Tuple[float, float], bands


Until Next Time!
SSEC