Fusion Data Framework Documentation

Release 0.0.0

John Schmitt, David R. Smith, Kevin Tritz, Howard Yuh

October 30, 2015

CONTENTS

CHAPTER

ONE

INTRODUCTION

1.1 About FDF

Fusion Data Framework (FDF) is a data access, management, and visualization framework for magnetic fusion experiments.

- Code repository
- Documentation
- Project documents
- Submit bugs or feature requests

Created by:

- John Schmitt, Princeton Plasma Physics Lab
- David R. Smith, U. Wisconsin-Madison
- Kevin Tritz, The Johns Hopkins U.
- Howard Yuh, Nova Photonics

1.2 About this documentation

FDF documentation is built using the Sphinx documentation generator for Python, and the source files are written in reStructuredText.

1.3 License

The MIT License (MIT)

Copyright (c) 2015 John Schmitt, David R. Smith, Kevin Tritz, Howard Yuh

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

CHAPTER

TWO

GETTING STARTED

2.1 User Guide

This guide is for people who want to use FDF on the PPPL Linux cluster. If you wish to contribute to the FDF project as a developer, see the developer guide.

HTML documentation is also available: http://fusion-data-framework.github.io/fdf/

To use FDF on the PPPL Linux cluster, load the module nstx/fdf (you may need to unload other nstx modules):

```
[sunfire06:~] % module load nstx/fdf

[sunfire06:~] % module list
Currently Loaded Modulefiles:
1) torque/2.5.2      5) idl/8.2           9) java/v1.6
2) moab/5.4.0        6) nstx/treedefs    10) nstx/mdsplus5
3) ppplcluster/1.1   7) nstx/epics       11) nstx/fdf
4) freetds/0.91      8) nstx/idldirs
```

Verify that Python points to the Anaconda distrubtion in the FDF project area:

```
[sunfire06:~] % which python
/p/fdf/anaconda/bin/python
```

If Python does not point to `/p/fdf/anaconda/bin/python`, then you likely loaded a different Python module that preempts the FDF/Anaconda distribution. If so, then you should unload the other Python module.

Next, start Python and verify that the Python path (`sys.path`) contains the Anaconda distrubtion at `/p/fdf/anaconda/lib/python2.7` and not other distributions such as `/usr/pppl/python/2.7.2/lib/python2.7`:

```
[sunfire06:~] % python
Python 2.7.10 |Anaconda 2.3.0 (64-bit)| (default, Sep 15 2015, 14:50:01)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org

>>> import sys
```

```
>>> from pprint import pprint
>>> pprint(sys.path)
```

If `sys.path` contains distributions other than `/p/fdf/anaconda/lib/python2.7`, then you likely loaded an additional Python module. If so, then you should unload the other Python module and restart Python.

Finally, you can import the FDF package:

```
>>> import fdf
```

See Usage Examples for more information about the usage and capabilities of FDF.

2.2 Developer Guide

This guide is for developers who want to contribute to the FDF project, and this guide describes the development workflow on the PPPL Linux cluster. If you simply want to use FDF on the PPPL Linux cluster, see the user guide.

HTML documentation is also available: http://fusion-data-framework.github.io/fdf/

First, configure your Linux cluster environment as described above in User Guide.

The FDF code repository is hosted on GitHub. To participate in the FDF project as a developer, you must create a GitHub account. The FDF project uses GitHub and Git for collaborative development and version control. You can submit bug reports and feature requests on the repository website.

Configure Git

On the PPPL Linux cluster, load the module git/1.8.0.2 (on Red Hat 6 systems, use git/2.4.2):

```
[sunfire08:~] % module avail git
--------------------- /usr/pppl/Modules/modulefiles --------------------
git/1.7.4.1(default)     git/1.8.0.2      git/2.4.2

[sunfire08:~] % module load git/1.8.0.2

[sunfire08:~] % module list
Currently Loaded Modulefiles:
1) torque/2.5.2      3) ppplcluster/1.1
2) moab/5.4.0        4) git/1.8.0.2
```

You may want to add the module load command to your shell start-up files: ~/.cshrc for csh/tcsh or ~/.bash_profile for bash.

Next, you must configure Git with your name and email (the same email associated with your GitHub account):

```
[sunfire08:~] % git config --global user.name "John Doe"
[sunfire08:~] % git config --global user.email "JohnDoe@email.com"
```

Also, we recommend setting an editor (e.g. vi, emacs, nedit) for Git comments:

```
[sunfire08:~] % git config --global core.editor nedit
```

You can inspect your Git configuration in the file ~/.gitconfig. For more information about Git configuration, see https://help.github.com/articles/set-up-git/ or https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup

Clone the FDF repository

Git clones repositories into a new directory in your current directory. In the right column of the FDF repo page (https://github.com/Fusion-Data-Framework/fdf), you can find the HTTPS URL (https://github.com/Fusion-Data-Framework/fdf.git) to clone FDF to your local directory

```
[sunfire08:~] % ls -d fdf
ls: fdf: No such file or directory

[sunfire08:~] % git clone https://github.com/Fusion-Data-Framework/fdf.git
Cloning into 'fdf'...
remote: Counting objects: 619, done.
remote: Total 619 (delta 0), reused 0 (delta 0), pack-reused 619
Receiving objects: 100% (619/619), 783.01 KiB, done.
Resolving deltas: 100% (279/279), done.

[sunfire08:~] % ls -d fdf
fdf/
```

Cloning via SSH is also feasible: https://help.github.com/articles/set-up-git/#next-steps-authenticating-with-github-from-git

Finally, add your new fdf directory to the `PYTHONPATH` environment variable:

```
[sunfire08:~] % setenv PYTHONPATH ${HOME}/fdf:$PYTHONPATH

[sunfire08:~] % echo $PYTHONPATH
/u/drsmith/fdf:<other directories>
```

You may want to add this action to your shell start-up files, as described above. In bash, use the export command to set `PYTHONPATH`.

Git workflow for FDF development

(1) Create a development branch (here, we call it devbranch) and checkout the new branch:

```
[sunfire08:~] % cd fdf

[sunfire08:~/fdf] % git branch
* master

[sunfire08:~/fdf] % git branch devbranch

[sunfire08:~/fdf] % git branch
devbranch
* master

[sunfire08:~/fdf] % git checkout devbranch
Switched to branch 'devbranch'
```

```
[sunfire08:~/fdf] % git branch
* devbranch
master
```

Devbranch initializes as a copy of master. `git branch` lists branches in your local repository, and the asterisk denotes the active branch. You can switch between local branches with `git checkout <LocalBranchName>`.

(2) Push devbranch to the remote FDF repository at GitHub (you may need to enter your GitHub username and password):

```
[sunfire08:~/fdf] % git push origin devbranch
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/Fusion-Data-Framework/fdf.git
 * [new branch]      devbranch -> devbranch
```

devbranch is now listed in the FDF repository at GitHub. `origin` is the alias for the remote GitHub repository. You can view your remote repositories and aliases with `git remote -v`.

(3) Proceed with FDF development within devbranch: commit changes, add/delete files, and push updates to GitHub.

As you complete small tasks, you should commit changes to your local repository with `git commit -a -m '<mymessage>'`. Also, each commit requires a short message describing the changes:

```
[sunfire02:~/fdf] % git commit -a -m 'added dictionary rows in logbook.py'
[devbranch bb6c58a] added dictionary rows in logbook.py
1 file changed, 16 insertions(+), 21 deletions(-)
```

If you do not specify a commit message with -m option, then Git will open your default editor and ask for a commit message (see Configure Git above). The -a option commits all file changes throughout the branch index, not simply your current directory. The branch index is the list of files Git tracks in the branch. `git commit -a` tracks changes to files in the branch index, so you must add new files to the index and remove deleted files from the index. You can view the branch index with `git ls-files`, and you can add new files to the index and remove deleted files from the index with `git add -A`:

```
[sunfire02:~/fdf] % touch temp.py

[sunfire02:~/fdf] % ls temp.py
temp.py

[sunfire02:~/fdf] % git ls-files temp.py

[sunfire02:~/fdf] % git add -A

[sunfire02:~/fdf] % git ls-files temp.py
temp.py
```

Note that temp.py appeared in the index only after the command `git add -A`. Similarly, deleted files stay in the index until the `git add -A` is given.

When you complete a large task, you should "push" changes to the devbranch on GitHub with `git push`:

```
[sunfire05:~/fdf] % git push origin devbranch
Counting objects: 10, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.30 KiB, done.
Total 6 (delta 3), reused 0 (delta 0)
To https://github.com/Fusion-Data-Framework/fdf.git
    129c5d9..a166825 devbranch -> devbranch
```

Again, "origin" signifies the branches on the remote GitHub repo.

(4) While you are working locally in devbranch, others may be modifying master at GitHub. When you are ready to merge devbranch into master, you should first merge the latest version of master from GitHub into your local devbranch. To retrieve the latest version of master from GitHub, use `git fetch`:

```
[sunfire05:~/fdf] % git fetch origin master
From https://github.com/Fusion-Data-Framework/fdf
* branch            master     -> FETCH_HEAD
```

Next, verify that you are in devbranch and merge origin/master into devbranch:

```
[sunfire08:~/fdf] % git branch
* devbranch
master

[sunfire05:~/fdf] % git merge origin/master
```

Next, push your local devbranch to devbranch on GitHub:

```
[sunfire05:~/fdf] % git push origin devbranch
```

Finally, on the GitHub website, in the devbranch area, submit a *pull request* to pull devbranch into master.

CHAPTER

THREE

USAGE EXAMPLES

First, import the FDF module:

```
>>> import fdf
```

3.1 Initiate a machine instance

Define a NSTX machine instance:

```
>>> nstx = fdf.Machine('nstx')
```

or pre-load a shotlist:

```
>>> nstx = fdf.Machine('nstx', [140000, 140001])
```

or pre-load an XP:

```
>>> nstx = fdf.Machine('nstx', xp=1013)
```

3.2 Load shots and XPs

Shots are added as referenced. For instance, without previous reference to 139980, you can enter:

```
>>> nstx.s139980.chers.plot()
```

Add shots to the NSTX instance:

```
>>> nstx.addshot(140000)
```

or a shotlist:

```
>>> nstx.addshot([141400, 141401, 141402])
```

or by XP:

```
>>> nstx.addshot(xp=1048)
```

or by date (string or int YYYYMMDD):

```
>>> nstx.addshot(date=20100817)
```

List shots presently loaded:

```
>>> dir(nstx)
```

or:

```
>>> nstx.listshot()
```

Get a custom **shotlist**:

```
>>> my_shotlist = nstx.get_shotlist(xp=1032)   # returns numpy.ndarray
```

CHAPTER

FOUR

GIT COMMAND REFERENCE

4.1 Tutorials

https://www.atlassian.com/git/tutorials

https://try.github.io/levels/1/challenges/1

https://help.github.com/articles/good-resources-for-learning-git-and-github/

4.2 Command summary

`git clone <URL>` - clone a local copy of a remote repo in a new directory
`git remote` - manage remote repo names (e.g. "origin") and URLs
`git fetch <remote>` - fetch updates from remote repo
`git push <remote> <branch>` - push current local branch to remote branch
`git merge <branch>` or `git merge <remote>/<branch>` - merge updates from local or remote branch into current local branch
`git branch` - manage branches in local and remote repos
`git checkout <branch>` - switch to a different local branch
`git add` - add files to the index
`git commit` - commit changes to the local branch
`git status` - view status of working directory
`git log` - view commit log of current local branch
`git diff` - view differences between working directory and index

4.3 Detailed command reference

4.3.1 Initiate or clone a repository

Clone a local copy of a repository (repo) in a new directory (remote repo is the "origin" remote):

```
% git clone <URL>
```

Initiate a repository in the current directory:

```
% git init
```

4.3.2 Interacting with the remote repo

List remotes and URLs (omitting -v lists only remote names, not URLs):

```
% git remote -v
```

Add a remote repo, such as "upstream" (good for the parent of a forked repo):

```
% git remote add <new-remote> <URL>
```

Fetch updates from a remote repo, such as "origin" (note that fetch does not merge anything into the local repo):

```
% git fetch <remote>
```

List remote branches:

```
% git branch -r
```

Merge remote branch into current branch:

```
% git merge <remote>/<branch>
```

Push local branch to remote repo:

```
% git push <remote> <branch>
```

Delete remote branch:

```
% git branch <remote> -d <branch>
```

4.3.3 Working within the local repo

List local branches:

```
% git branch
```

Create new branch from master branch in local repo:

```
% git branch <new branch>
```

Create new branch from existing branch in local repo:

```
% git branch <new branch> <existing branch>
```

Delete local branch:

```
% git branch -d <branch>
```

Checkout (switch) to different local branch (or create local version of remote branch if local branch does not exist):

```
% git checkout <branch>
```

Merge local branch into current branch:

```
% git merge <branch>
```

Update index with any new/deleted/moved files:

```
% git add -A
```

Commit changes in current branch:

```
% git commit -a -m "message"
```

View differences with index:

```
% git diff
```

View differences with another branch:

```
% git diff <target branch>
```

View commit log for cirrent branch:

```
% git log
```

View status of working directory:

```
% git status
```

CHAPTER

FIVE

PACKAGE REFERENCE

5.1 Package overview

FDF is a data access/management framework for magnetic fusion experiments.

Modules

- `factory` - root module for FDF package
- `fdf_globals` - package-wide constants
- `fdf_signal` - signal class module
- `fdf/modules/` - diagnostic sub-modules.

Usage

```
>>> import fdf
>>> nstx = fdf.Machine('nstx')
>>> nstx.s140000.logbook()
>>> nstx.addshots(xp=1048)
>>> nstx.s140001.mpts.plot()
```

5.2 Module factory.py

Root module for the FDF package.

Classes

- Machine - root class for the FDF package
- Shot - shot container class
- Logbook - logbook connection class
- Container - diagnostic container class
- Node - mdsplus signal node class

5.3 Class factory.Machine

class factory.**Machine**(*name='nstx'*, *shotlist=[]*, *xp=[]*, *date=[]*)

Factory root class that contains shot objects and MDS access methods.

Note that fdf.factory.Machine is exposed in fdf.__init__, so fdf.Machine is valid.

Usage:

```
>>> import fdf
>>> nstx = fdf.Machine('nstx')
>>> nstx.s140000.logbook()
>>> nstx.addshots(xp=1048)
>>> nstx.s140000.mpts.plot()
>>> nstx.listshot()
```

Machine class contains a model shot object: nstx.s0

Shot data can be accessed directly through the Machine class:

```
>>> nstx.s141398
>>> nstx.s141399
```

Alternatively, a list of shot #'s may be provided during initialization:

```
>>> nstx = Machine(name='nstx', shotlist=[141398, 141399])
```

Or added later using the method addshot():

```
>>> nstx.addshot([141398, 141399])
```

addshot(*shotlist=[]*, *date=[]*, *xp=[]*, *verbose=False*)

Load shots into the Machine class

Usage

```
>>> nstx.addshot([140000 140001])
>>> nstx.addshot(xp=1032)
>>> nstx.addshot(date=20100817, verbose=True)
```

Note: You can reference shots even if the shots have not been loaded.

5.4 Class factory.Shot

class factory.**Shot**(*shot*, *root=None*, *parent=None*)

5.5 Class factory.Logbook

class factory.**Logbook**(*name='nstx'*, *root=None*)

5.6 Module fdf_signal.py

fdf-signals.py - module containing Signal class

Classes

- Signal - signal class for data objects

5.7 Class fdf_signal.Signal

class fdf_signal.**Signal**(***kwargs*)

sig=fdf.Signal(signal_ndarray, units='m/s', axes=['radius','time'], axes_values=[ax1_1Darray, ax2_1Darray], axes_units=['s','cm']

e.g.: mds.Signal(np.arange((20*10)).reshape((10,20)), units='keV', axes=['radius','time'], axes_values=[100+np.arange(10)*5, np.arange(20)*0.1], axes_units=['s','cm'])

or an empty signal: s=mds.Signal() default axes order=[time, space] sig=fdf.Signal(units='m/s', axes=['radius','time'], axes_values=[radiusSignal, timeSignal])

5.8 Module fdf_globals.py

Package-level attributes, methods, and FdfError class

5.9 Module tests.py

5.10 Class tests.TestShotFixture

class tests.**TestShotFixture**(*methodName='runTest'*)

A test fixture to validate the data structure in shot objects

setUp()

Setup method for all test cases in this text fixture

testShotCase()

A test case for shot objects to ensure:

•Every containter contains at least a signal or sub-container

•Every signal contains at least 1 axis

•Every axis is listed in signal.axes

•Every item is signal.axes is a valid axis object

•Every signal possesses a valid plot method

TODO: Ensure signal axes are consistent with signal.axes

CHAPTER

SIX

PROJECT DOCUMENTS

October 9, 2015 meeting

A

F

L

M

S

T