Fusion Data Framework Documentation

Release 0.0.0

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

October 08, 2015

CONTENTS

1 Introduction **1**
- 1.1 About FDF . . . 1
- 1.2 Project Documents . . . 1
- 1.3 License . . . 1

2 Getting Started Guides **3**

3 Usage Examples **4**
- 3.1 Define a machine instance . . . 4
- 3.2 Loading shots and XPs . . . 4

4 Package Reference **6**
- 4.1 FDF Package . . . 6
- 4.2 Module factory.py . . . 6
- 4.3 Class factory.Machine . . . 7
- 4.4 Class factory.Shot . . . 7
- 4.5 Class factory.Logbook . . . 7
- 4.6 Module fdf_signal.py . . . 8
- 4.7 Class fdf_signal.Signal . . . 8
- 4.8 Module fdf_globals.py . . . 8

Python Module Index **9**

Index **10**

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: https://github.com/Fusion-Data-Framework/fdf

HTML Documentation or PDF Documentation

Project Documents

Submit bugs or feature requests: https://github.com/Fusion-Data-Framework/fdf/issues

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 Project Documents

1.2.1 October 9, 2015 meeting

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 GUIDES

CHAPTER

THREE

USAGE EXAMPLES

Import the FDF module:

```
>>> import fdf
```

3.1 Define 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 Loading shots and XPs

Add shot(s) 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 (YYYYMMDD (string or int)):

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

List shots presently loaded:

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

Get a custom shotlist:

```
>>> my_shotlist = nstx.get_shotlist(xp=1032)
>>> type(my_shotlist)
<type 'numpy.ndarray'>
```

CHAPTER

FOUR

PACKAGE REFERENCE

4.1 FDF Package

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()
```

4.2 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

4.3 Class factory.Machine

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

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

Usage:

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

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 addshot method:

```
>>> 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.

4.4 Class factory.Shot

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

4.5 Class factory.Logbook

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

4.6 Module fdf_signal.py

fdf-signals.py - module containing Signal class

Classes

- Signal - signal class for data objects

4.7 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])

4.8 Module fdf_globals.py

Package-level constants and FdfError class

- genindex

f

factory, 6
fdf.__init__, 6
fdf_globals, 8
fdf_signal, 8

A

addshot() (factory.Machine method), 7

F

factory (module), 6
fdf.__init__ (module), 6
fdf_globals (module), 8
fdf_signal (module), 8

L

Logbook (class in factory), 7

M

Machine (class in factory), 7

S

Shot (class in factory), 7
Signal (class in fdf_signal), 8