# InSet Documentation

***Release 1***

**Rahul Singh**

July 20, 2015

**This website provides the documentation for the usage and extension of the INstrumentSETtings beam instrumentation toolbox**

This is not in the above paragraph?

CHAPTER

# ONE

# CONTENTS

## 1.1 Beam

This module defines the beam class

The module takes the following arguments

Property — Key — Value Type — Remarks

Particle type — par_type — String — Ion type (p, U, Ar etc.)

Charge state — charge_state — Integer — Charge state

Atomic mass — atomic_mass — Integer — 2 for Hydrogen

Particle energy — kin_energy — Float — Kinetic energy per nucleon

Particle number —par_num — Integer — Total number of particles (ions)

Distribution type — d_type — String — a for arbitrary, p for parabolic, g for gaussian and kv for KV distribution

X Distribution —- x_dist — List of integers for 'a', two Ints for parabolic and gaussian — Phase space distribution in x plane

Y Distribution —- y_dist — Same as X Dist. — Phase space distribution in y plane

Z Distribution —- z_dist — Same as X Dist. — Phase space distribution in z/s plane

> par_type=None, charge_state= None, atomic_mass = None, par_num= None, d_type = 'ggg', x_dist = [0,5], y_dist =[0,5], z_dist =[0,100]

The arguments can be passed in this order or by defining a dictionary or by calling a file where the parameters exist

**class** beam.**dynamicbeam**(**args*, ***kwargs*)

> This attributes are similar to a static beam, however, the beam energy, number of beam particles and beam structure is updated for several turns (depends on the length of list) and stored
>
> **gaussian**(*x*, *mu*, *sig*)
>
> > Returns the normal distribution with x number of points, and mean mu, and sig
>
> **listfiles**()
>
> > THis function will list all the static and dynamic beams
>
> **parabolic**(*x*, *xmin*, *xmax*)
>
> > Returns the normal distribution with x number of points, and mean mu, and sig
>
> **plot**()
>
> > This function will plot the profile of the beam in the mentioned axis

**save**(*name_of_file*, *description*)
This function will save the beam object to an external file in the directory called "defined_beams" in the source directory

**structure**()
This function defines the structure of the beam based on the beam parameters

**class** beam.**staticbeam**(**args*, ***kwargs*)
Beam class defines the static beam object

It creates a beam object instance the parameters in a special order are specified, or simply by passing a beam dictionary

A save keyword 's' can be used to save the beam object in a file, which can be loaded later

**gaussian**(*x*, *mu*, *sig*)
Returns the normal distribution with x number of points, and mean mu, and sig

**classmethod listfiles**(*staticbeam*)
THis function will list all the static and dynamic beams

**load**(*name_of_file*)
This function will load the beam object from the specified file in the directory called "defined_beams" in the source directory

**parabolic**(*x*, *xmin*, *xmax*)
Returns the normal distribution with x number of points, and mean mu, and sig

**plot**()
This function will plot the profile of the beam in the mentioned axis

**save**(*name_of_file*, *description*)
This function will save the beam object to an external file in the directory called "defined_beams" in the source directory

**structure**()
This function defines the structure of the beam based on the beam parameters

## 1.2 Indices and tables

- genindex
- modindex
- search
- Home
- *Table of contents*
- *Table of page*

## 1.3 Machine

This module defines the beam object

The module takes the following arguments

Property — Key — Value type — Description

Circumference — circumference — Float — Circumference of the machine

Compaction factor — com_fact — Float — Momentum compaction factor

Set tune — set_tune — List of Float — Horizontal and vertical tune

Set Chromaticity — set_chro — List of float — Horizontal and vertical chromaticity

**class** machine.**dynamicmachine**(**args*, ***kwargs*)
This attributes are similar to a static beam, however, the beam energy, number of beam particles and beam structure is updated for several turns (depends on the length of list) and stored

**save**(*name_of_file*, *description*)
This function will save the beam object to an external file in the directory called "defined_beams" in the source directory

**class** machine.**staticmachine**(**args*, ***kwargs*)
The machine class defines all the machine parameters

**save**(*name_of_file*, *description*)
This function will save the beam object to an external file in the directory called "defined_beams" in the source directory

### 1.3.1 Indices and tables

- genindex
- modindex
- search
- Home
- *Table of contents*
- *Table of page*

Use the tutorial here to learn about Sphinx: *[SPHINXDOC]*.

## 1.4 Device Modules

Each Diagnostic sensor description and settings are documented here.

### 1.4.1 Current Transformers

Generic Transformer object  Generic Trafo module takes beam and machine object and returns the TrafoOut

The module takes the following arguments

Beam — Beam object fully specifying the beam

Machine — Accelerator setting object

TrafoType (Optional) — Specific transformer types to define exact Trafo behaviour

**class** TrafoModule.**generictrafo**(**args*, ***kwargs*)
The Generic trafo class defines all the generic trafo parameters

**combine_systems**(**args*, ***kwargs*)
Combines all the subsystems to the sensor

**load**(*name_of_file*)
Loads the specific trafo settings

**optimize**(*observable_in*, *constraints*)
Predicts optimal settings for all the variable parameters in the device module, common module or any combination

**output**(*observable_in*)
Calculates the output of the device module or the system

**save**(*name_of_file*, *description*)
This function will save the beam object to an external file in the directory called "defined_beams" in the source directory

### 1.4.2 Add functions from Python library

```
io.open()
```

### 1.4.3 Indices and tables

- genindex
- modindex
- search
- Home
- *Table of contents*
- *Table of page*

## 1.5 Common Modules

The common modules consist of electronics, optics systems cables etc. They are described and documented here.

### 1.5.1 Amplifiers and Attenuators

Generic amplifier and attenuator definition  Generic Amplifier Module

The module takes the following arguments

Amplification — The amplification/attenuation in (dB)

Noise figure — Accelerator setting object

Input Noise — When the input is open or terminated (in nV/sqrt(Hz))

AmplifierType (Optional) — Specific amplifier implementation

**class** AmpAttModule.**genericAmpAtt**(**args*, ***kwargs*)
The Generic trafo class defines all the generic trafo parameters

**save**(*name_of_file*, *description*)
This function will save the beam object to an external file in the directory called "defined_beams" in the source directory

### 1.5.2 Lenses

### 1.5.3 Indices and tables

- genindex
- modindex
- search
- Home
- *Table of contents*
- *Table of page*

CHAPTER

TWO

# INDICES AND TABLES

- genindex
- modindex
- search

Use the tutorial here to learn about Sphinx: *[SPHINXDOC]*.

[SPHINXDOC] This is Sphinx doc documentation –> http://sphinx-doc.org/latest/tutorial.html.

## a



## b



## m



## t