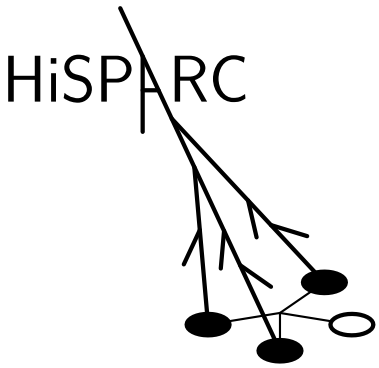

Public Database Documentation

Release 0.1

Arne de Laat

March 26, 2013

CONTENTS

The HiSPARC Public Database is the interface through which everyone can access the data and our station administration is done.

Contents:

CHAPTER

ONE

API REFERENCE

Application Programming Interface for HiSPARC Public Database

The API simplifies data access for data contained in the HiSPARC Public Database. It was born out of the need for easy access to up-to-date information about stations.

The following packages and modules are included:

urls urls that can be called and will be passed on to functions

views definitions that return specific data from the database

Contents:

1.1 API Views Reference

django_publicdb.api.views.**clusters**(*request*, *country_id=None*)

Get cluster list

Retrieve a list of all clusters or only the clusters in a specific country. By cluster we here mean the main clusters, which contain subclusters.

Parameters **country_id** – a country number identifier, give this to only get clusters from a specific country.

Returns clusters. A list containing the name and number of all clusters that matched the given parameters.

django_publicdb.api.views.**config**(*request*, *station_id*, *year=None*, *month=None*, *day=None*)

Get station config settings

Retrieve the entire configuration of a station. If no date if given the latest config will be sent, otherwise the latest on or before the given date.

Parameters

- **station_id** – a stationn number identifier.
- **year** – the year part of the date.
- **month** – the month part of the date.

- **day** – the day part of the date.

Returns config. A dictionary containing the entire configuration from the HiSPARC DAQ.

django_publicdb.api.views.**countries**(*request*)
Get country list

Retrieve a list of all countries with active stations.

Returns countries. A list containing the name and number of all countries.

django_publicdb.api.views.**get_cluster_dict**(*country=None*)

django_publicdb.api.views.**get_country_dict**()

django_publicdb.api.views.**get_station_dict**(*subcluster=None*)

django_publicdb.api.views.**get_subcluster_dict**(*cluster=None*)

django_publicdb.api.views.**has_data**(*request*, *station_id*, *year=None*, *month=None*, *day=None*)
Check for presence of cosmic ray data

Find out if the given station has measured shower data, either on a specific date, or at all.

Parameters

- **station_id** – a stationn number identifier.
- **year** – the year part of the date.
- **month** – the month part of the date.
- **day** – the day part of the date.

Returns has_data. A boolean, True if the given station has shower data, False otherwise.

django_publicdb.api.views.**has_weather**(*request*, *station_id*, *year=None*, *month=None*, *day=None*)
Check for presence of weather data

Find out if the given station has measured weather data, either on a specific date, or at all.

Parameters

- **station_id** – a stationn number identifier.
- **year** – the year part of the date.
- **month** – the month part of the date.
- **day** – the day part of the date.

Returns has_weather. A boolean, True if the given station has weather data, False otherwise.

django_publicdb.api.views.**json_dict**(*dict*)
Create a json HTTPResponse

django_publicdb.api.views.**man**(*request*)
Give overview of the possible urls

django_publicdb.api.views.**num_events**(*request*, *station_id*)
Get total number of events for a station

Retrieve the number of events that a station has measured during its entire operation. The following functions each dig a little deeper, going for a shorter time period.

Parameters **station_id** – a stationn number identifier.

Returns num_events. A number (long) containing the total number of events ever recorded by the given station.

django_publicdb.api.views.**num_events_day**(*request*, *station_id*, *year*, *month*, *day*)
Get total number of events for a station on a given date

Retrieve the total number of events that a station has measured on a date.

Parameters

- **station_id** – a station number identifier.
- **year** – the year part of the date.
- **month** – the month part of the date.
- **day** – the day part of the date.

Returns num_events. A number (long) containing the number of events recorded by the station on the given date.

django_publicdb.api.views.**num_events_hour**(*request*, *station_id*, *year*, *month*, *day*, *hour*)
Get number of events for a station in an hour on the given date

Retrieve the total number of events that a station has measured in that hour.

Parameters

- **station_id** – a station number identifier.
- **year** – the year part of the date.
- **month** – the month part of the date.
- **day** – the day part of the date.
- **hour** – the hour for which the number of events is to be retrieved.

Returns num_events. A number (long) containing the number of events recorded by the station in hour on date.

django_publicdb.api.views.**num_events_month**(*request*, *station_id*, *year*, *month*)
Get total number of events for a station in the given month of a year

Retrieve the total number of events that a station has measured during the given month.

Parameters

- **station_id** – a station number identifier.
- **year** – the year in which to look for the month.
- **month** – the month for which the number of events is to be given.

Returns num_events. A number (long) containing the total number of events recorded by the given station in the given month of the given year.

django_publicdb.api.views.**num_events_year**(*request*, *station_id*, *year*)
Get total number of events for a station in the given year

Retrieve the total number of events that a station has measured during the given year.

Parameters

- **station_id** – a station number identifier.
- **year** – the year for which the number of events is to be given.

Returns num_events. A number (long) containing the total number of events recorded by the given station in the given year.

django_publicdb.api.views.**station**(*request*, *station_id*)
Get station info

Retrieve important information about a station.

Parameters **station_id** – a station number identifier

Returns station_info. This is a dictionary containing info about the station. Most importantly, this contains information about the position of the station, including the position of the individual scintillators.

django_publicdb.api.views.**stations**(*request*, *subcluster_id=None*)
Get station list

Retrieve a list of all stations or all stations in a subcluster.

Parameters **subcluster_id** – a subcluster number identifier. If given, only stations belonging to that subcluster will be included in the list.

Returns stations. This is a list containing dictionaries which consist of the name and number of each station (matching the subcluster).

django_publicdb.api.views.**stations_with_data**(*request*, *year*, *month*, *day*)
Get stations with data

Retrieve a list of all stations which have data on the given date.

Parameters

- **year** – the year part of the date.
- **month** – the month part of the date.
- **day** – the day part of the date.

Returns stations. A list containing the name and number of each station that has measured events on the given date.

django_publicdb.api.views.**stations_with_weather**(*request*, *year*, *month*, *day*)
Get stations with weather data

Retrieve a list of all stations which have weather data on the given date.

Parameters

- **year** – the year part of the date.
- **month** – the month part of the date.
- **day** – the day part of the date.

Returns stations. A list containing the name and number of each station that has measured weather data on the given date.

django_publicdb.api.views.**subclusters**(*request*, *cluster_id=None*)
Get subcluster list

Retrieve a list of all subclusters or all subclusters in a specific cluster.

Parameters **cluster_id** – a cluster number identifier, give this to only get subclusters from this cluster.

Returns subclusters. A list containing the name and number of all subclusters that matched the given parameters.

django_publicdb.api.views.**validate_date**(*date*)
Check if date is outside HiSPARC project range

If not valid, a 404 (Not Found) should probably be returned to the user.

Contents:

CHAPTER

TWO

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

d

INDEX