Extracting Data from SUFO

Patricio F. Ortiz

1 Tools to access data

The Sheffield Urban Flows Observatory ("the observatory" from here onwards) acts as a data repository for several data streams covering Air Quality, Meteorological conditions and traffic data across Sheffield and its immediate vicinity.

In this document, details about the data collection and internal operations of **the observatory** are not relevant. In this document, we shall focus on **how** to explore and extract data from the observatory. We assume that users of these tools have some experience working with scientific/engineering real-life data. We designed these tools with research purposes in mind.

Data from different providers and collaborators come in various formats and styles. One of the essential roles of the observatory is to provide its users with a uniform system.

We created two web-based tools to provide a uniform data retrieval service.

The Portal is a graphic interface which lets users see the location of the sensors and extract small amounts of data from selected sensors. The portal also allows a quick view of the conditions of a given quantity across an area of the city, for instance, traffic flow, or NO concentration.

The local portal (restricted access to university members) is
`https://ufdev21.shef.ac.uk/sufobin/sufoPortal`

The public, open portal is
`https://sheffield-portal.urbanflows.ac.uk/uflobin/sufoPortal`

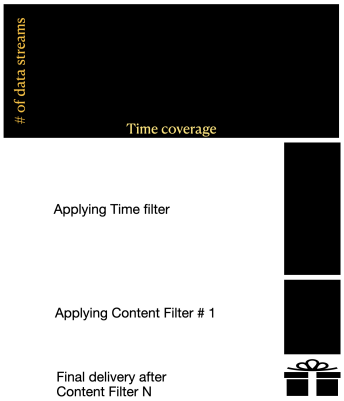
The Data Extraction Tool (**DXT**) is a web-based tool capable of extracting specific data from the observatory. It also provides a basic initial web interface for human training and extraction. However, its primary purpose is to act as a data provider for automated systems, achieved through its API approach. In other words, bots may regularly retrieve data from the observatory.

The local DXT (restricted access to university members) is
`https://ufdev21.shef.ac.uk/sufobin/sufoDXT`

The public, open Data Extraction Tool is
`https://sheffield-portal.urbanflows.ac.uk/uflobin/sufoDXT`

It should be noted that the portal uses the DXT to extract data and serve it on the browser. Extracting data from the portal is possible, but it requires human intervention; it is a convenient way to get acquainted with the system.

2 Getting the desired data



The observatory acts very much like an archiving system; hence, data extraction is time oriented. When either tool is invoked without specifying anything, as shown in the URLs above (simplest form), key parameters are given default values.

The extraction's final time is set to the current time, while the extraction's initial time is set to be "3 hours ago".

Both tools return all data streams when default values are used. This may still be too much for users. API key/value pairs allow users to both define data filters and specify extraction parameters and output formats.

All filters are described in detail in the API appendix.

2.1 Learn from the demos & examples in the API description

A close examination of the opening page of DXT gives a good hint of the various available filters.

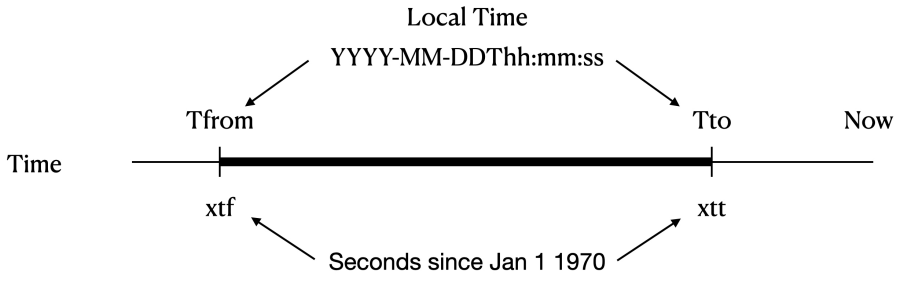
The SHARE-URL button picks up all the options you have entered in the web form and produces a URL with the correct syntax.

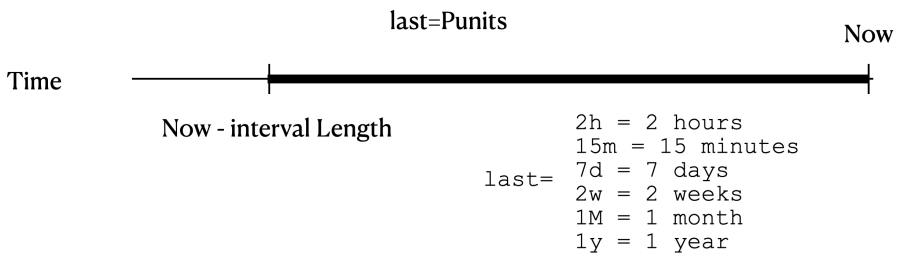
In addition, the **portal** acts as a data extraction demonstrator. This happens when one generates a plot and clicks the GET-DATA button in the plot window, that action invokes the DXT to extract just the data which is needed.

Throughout this document, the texts shown in boxes are actual/active links that allow us to illustrate points.

2.2 Time Filters:

Two points define every data extraction in the timeline: **initial-time** and **final-time**, as shown in the figure:





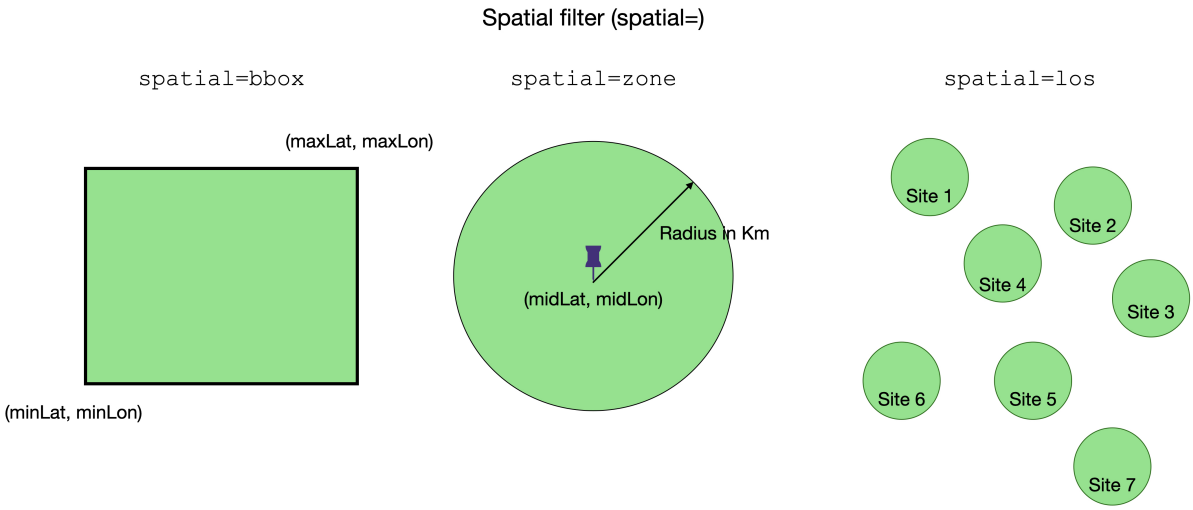

```
last=  2h = 2 hours
       15m = 15 minutes
       7d = 7 days
       2w = 2 weeks
       1M = 1 month
       1y = 1 year
```

The figure illustrates three possible ways to specify the initial and final time points: a) using a human-readable form **(Tfrom, Tto)**, b) using Unix time (seconds since 1970.0) **(xtf, xtt)** or c) using **last=Punits** to indicate an interval which ends "now".

The last 3 hours of data will be extracted if no time is specified,

2.3 Location filters:

There are a few ways to apply restrictions to the location of the sensors: a) sites located within a circle centred at a point (Lon, Lat) of radius in Km. b) sites within a bounding box, where min-Latitude, max-Latitude, min-Longitude and max-Longitude are specified. c) is a variation of (a), where the locations of a list of sites define one or more extraction areas. d) it's possible to filter sites by their height above sea level. Finally, it is possible to specify a case-sensitive pattern which the site's address must contain (when the street address is present)



2.4 Content filters:

Data is organised by several categories, and each category can be used as a filter to choose a single detector or a set of detectors from which to extract data. Data collected can originate from our sensors or external sensors.

`siteSets` are sets of sites associated with a given project. `ufloSites` is the denomination of all data collected from the observatory's deployed instruments. `SCC_sites` groups traffic data collected from the Sheffield City Council (SCC).

Sometimes one is interested in data collected by specific sites or sensors or a combination of sites and sensors (what we call pairs). Filters exist for that purpose.

It is also possible to apply a filter by quantity measured, regardless of its origin, for instance, retrieves all measurements of air temperature.

Other temporal filters allow retrieving data from specific days of the week, specific months or even ranges in "time of the day".

2.5 Data-volume control

Various instruments record measurements at different rates, and the observatory has no control on that. Some instruments report readings every minute, that is, 1440 readings per day, while others do less often.

By default, the observatory returns every single observation recorded for each instrument, but this may produce more data than users require. The API contains a key which controls the amount of readings returned to users: `freqInMin`.

`freqInMin`	Effect	Active-Links
1 (default)	Bring everything	sufoDXT?freqInMin=1
5	Pick data when mod(min,5) = 0	sufoDXT?freqInMin=5
15	Pick data when mod(min,15) = 0	sufoDXT?freqInMin=15
30	Pick data when mod(min,30) = 0	sufoDXT?freqInMin=30

Users should consider using this option to reduce the returned data-volume in situations when a long time-span needs to be examined and full temporal resolution is not necessary.

The values returned represent **values measured when the number of minutes is a multiple of the requested frequency**. We **do not perform an average over the interval**.

> Users who need to use an average over a certain number of minutes should define `freqInMin=1` to return every data point and perform the averaging themselves.

2.6 Data output: formats and delivery

The observatory serves the data in two formats: CSV (default) and JSON. The output may be individual tables for a given sensor or one table for sensors in a given datastream/family. To illustrate this point, the observatory contains data from more than 600 traffic sensors. It may be more desirable to retrieve 1 table containing data for all sensors rather than 600+ individual tables returned by default. Users may have different needs; hence these options exist.

CSV offers the option to get tables with a detailed description of their column content, including the location of the sensor and other information, but as there is no standard way to do this, the observatory also offers the chance to obtain bare CSV tables, with the first row representing the name of each column. In bare format, these tables can be fed directly to Matlab or Pandas (in Python).

sufoDXT?fmt=csv&tabCont=rich CSV table with rich description of each column (default).

sufoDXT?fmt=csv&tabCont=minimal CSV table with compact description of each column.

sufoDXT?gdata=byPairId CSV when data is grouped by site+sensor pairs.

sufoDXT?gdata=nogroup CSV when data is grouped by sensor families, producing fewer tables.

JSON files offer the option of column oriented or row oriented data storage.

sufoDXT?fmt=jsoncols JSON output grouping quantities by "column".

sufoDXT?fmt=jsonrows JSON output grouping quantities by "row". This kind of output is larger than the option by columns, but it is a format more "human readable".

JSON is of course more bulky than CSV, but it is a good exchange format if what gets the data is an automated system.

JSON format always shows data on a sensor by sensor basis.

By default, the results go to the browser (as in the previous examples). However, it's also possible to save a table directly into a file (the browser will show users the option to save it). A more efficient way to retrieve data involves using **zip**. In that case, DXT returns a **zip file** containing a folder with individual files for each of the tables retrieved.

sufoDXT?output=browser sufoDXT?output=saveas sufoDXT?output=zip

The zip option is the best when a large volume of data needs to be retrieved, and it may be the most convenient one for automated retrieval mode, although we have tried CSV and JSON formats on the browser to connect external services to our data services, and they work without any problem.

3 Data Output: optional extra information

The observatory stores time as seconds from 1970, which is a perfectly fine format for machines, but humans may need more readable formats to make sense of the outputs. It is possible then to append several optional columns to the data: full timestamps in local time (or Universal time), date of the observation, time of the observation (local time), individual columns for year, month, day, hour minute and second, minutes since midnight, time of the day (HHMM) format, day of the week (numerical or two character strings) or Unix time in milliseconds.

It is also possible to add columns for geolocation, site+sensor (pairID) and other time-related quantities.

Examples of these options are given in the API section.

4 Quality Control indicators

Sensor data is far from perfect, and it can sporadically or consistently produce erroneous data.

Identifying these cases where dubious data occur is essential, and it is up to the users to weigh these values appropriately when using data from SUFO.

To make this task easier, we created a series of analyses to spot dubious data points. We can assign a flag to each observation, or to replace suspicious values (including no data) with values which make them

be easily recognisable by dedicated analysis software.

The API key `qcopt` allows users to decide how to treat data with quality issues, and it can take the following values:

pruneData – the default option– the action is to replace missing values and values which are below or above the expected physical values with dedicated patterns. The table below shows the default values assigned for three specific cases:

Kind of problem	API key	Default value
Invalid or missing data	`udfnoval`	-32768
Value below expected range	`udfbelow`	-32769
Value above expected range	`udfabove`	-32767

sufoDXT?qcopt=prunedata&udfbelow=:-(&udfnoval=NaN

showincolumns In this case, a QC column is added after the quantity column to indicate whether a QC error has occurred. The QC values are a binary composite (or'ed) of various components. A value of 0 means our methods did not detect any issue, which does not mean that other problems may have escaped our scrutiny.

Flag meaning	Flag value	Flag meaning	Flag value
No issue	0	Below lower limit	8
No data	1	Above upper limit	16
Too early	2	In sharp Peak	32
Too late	4	In sharp dip	64
		In flat area	128

sufoDXT?qcopt=showincolumns

ignoreqc In this case, no QC is performed on the data. Users should understand that it is now up to them to spot anomalies in the data and to avoid feeding them blindly to their own procedures.

sufoDXT?qcopt=ignoreQC

Note that the default value for no-value cases is -32768

5 API details

The two tools being webapps, have an API consisting of key-value pairs, in the form `key=value` or `key=value1,value2,value3`

The **value** field may represent a single value or a comma separated list of values (**CSL**), a method used to indicate that if one of the values in the list matches, then whatever is described is selected. In most cases the match is expected to be identical, but in some cases patterns can be used (partial matching) or even negative matches can be specified.

If any of the values in the CSL contains an asterisk (`*`), it is assumed that users need the pattern to contain the value anywhere in the matching component. For instance, one can say `byFamily=SCC_*` to select all datastreams related to traffic operated by the Sheffield City Council.

If any of the values in the CSL contains the exclamation point (`!`), it is assumed that the user does not value anything which matching that particular value or pattern. For instance, `byFamily=!SCC_*` picks up data from any family which does not contain traffic data from the Sheffield City Council.

In the examples below, the elements in boxes are actual links which will perform the operations for you to examine the outputs and options.

5.1 Defining the extraction interval

`key=value` value is a single value, no CSL.

Tfrom=YYYY-MM-DDThh:mm:ss The initial time of the extraction interval in ISO8601 notation in Local Time.

Tto=YYYY-MM-DDThh:mm:ss The final time of the extraction interval in ISO8601 notation in Local Time. Live examples follow:

sufoDXT?tFrom=2022-07-01T00:00:00&tTo=2022-07-08T23:59:59 or

sufoDXT?tFrom=2022-07-01&tTo=2022-07-08

xtf=Unix-time-stamp The initial time of the extraction interval expressed in seconds since Jan 1, 1970

xtt=Unix-time-stamp The final time of the extraction interval expressed in seconds since Jan 1, 1970

last=P-units Used to extract information for the last **N u** units. N is a number, u can be "h" for hours, "d" for days, "m" for minutes, "M" for months, "w" for weeks and "y" for year

sufoDXT?last=6h retrieve the last 6 hours of data

sufoDXT?last=3d retrieve the last 3 days of data

sufoDXT?last=2w retrieve the last 2 weeks of data

5.2 Other temporal filters

byDOW Indicates a list of days of the week which should be extracted.

Valid values are: `Mon, Tue, Wed, Thu, Fri, Sat, Sun`

sufoDXT?tFrom=2022-07-01&tTo=2022-09-30&byDow=Sat,Sun retrieves data for Saturdays and Sunday between July 1 2022 and September 30, 2022

byMonth Month of the year: (3-letter abbreviation)

byTOD Time of day, expressed as HHMM::HHMM. HHMM goes from 0000 to 2359. This can be a comma separated list defining intervals of interest within days.

sufoDXT?tFrom=2022-09-20&tTo=2022-09-30&byTOD=0500::0800,1600::1930

retrieves data between July 2 2022 and September 30, 2022 between the hours of 5am and 8am as well as between 16:00 and 19:30

byYear Filter data for a given year only. Although it is not recommended to extract that much data.

Of course, the longer the extraction interval, and the larger the set of sites to retrieve, the longer it will take to retrieve data from the observatory, occasionally causing the browser to freeze or crash.

5.3 Content Filters

byFamily=CSL selects data by datastream identifiers. Supports '*' and '!' notation

sufoDXT?byFamily=eWatch Pull data from one set of sensors

sufoDXT?byFamily=eWatch,AMfixed Pull data from two sets of sensors

sufoDXT?byFamily=SCC_* Pull data from all traffic sensors.

sufoDXT?byFamily=!SCC_* Pull data not from traffic data sensors

bySitesSet=CSL select data if it belongs to a family of sites, as defined by the observatory

sufoDXT?bySitesSet=ufloSites Pull data from SUFO operated sites

sufoDXT?bySitesSet=SCC_sites Pull data from traffic data sites operated by Sheffield City Council

bySite=CSL select data for a list of sites only.

sufoDXT?bySite=S0001 Pull data from one site

byContent=CSL select data which contains sensors which measure given quantities and show every quantity that sensor measures.

sufoDXT?byContent=MET_TEMP Pull data from sites where temperature is measured.

bySelect=CSL Select sensors which contain certain quantities and ignores other measured quantities in the output tables.

sufoDXT?bySelect=MET_TEMP Pull data from sites where temperature is measured and only show AirTemperature

bySensor=CSL select data coming from a particular sensor list of sensorIDs)

sufoDXT?bySensor=711,2450204 Pull data from two sensors.

5.4 Location Filters

byNature=value select data which has been labeled as "outdoors" or "indoors".

spatial=value when used, indicated the mode for spatial extraction. Possible values are:

"**zone**" (circular zone of radius **radKm** centred at **(midLat, midLon)**

sufoDXT?spatial=zone&midLon=-1.501559&midLat=53.367129&radKm=0.5

"**bbox**" zone delimited by 2 parallels and 2 meridians defined by (minLat, maxLat) and (minLon, maxLon)

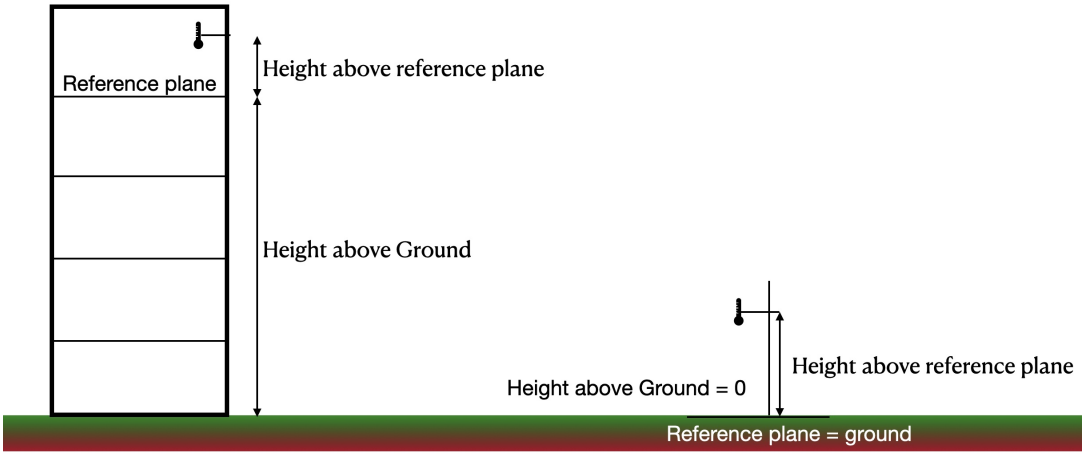
sufoDXT?spatial=bbox&minLat=53.3745&maxLat=53.390&minLon=-1.472&maxLon=-1.4355

"**los**" circular zones of radius **radKm** centered in sites listed using **bySiteLoc**

sufoDXT?spatial=los&bySiteLoc=S0001,S0047,S0055&radKm=1.2

Filtering by range of altitudes minimum and maximum value of height above sea level (in metres) can be specified and sites with altitude within the range are selected. Both limits need to be specified. Required: **(min_hasl, max_hasl)**

sufoDXT?min_hasl=80&max_hasl=100 get sites between 80 and 100 metres above sea level.



Filtering by height above ground minimum and maximum value of height above ground (in metres) can be specified. This refers to the height of a fiducial horizontal reference plane above the ground. Both limits need to be specified. Required: **(min_hag, max_hag)**

sufoDXT?min_hag=0&max_hag=5 get sites whose reference plane is between 1 and 5 metres above ground.

Filtering by height above reference level minimum and maximum value of height above reference level (in metres) can be specified. This refers to the instrument's height above the horizontal reference plane. Both limits need to be specified. Required: **(min_haref, max_haref)**

sufoDXT?min_haref=0&max_haref=5 get sites whose reference plane is between 1 and 5 metres above ground.

byAddress=pattern select data whose address contains a given pattern. Pattern is case sensitive and it does not require the presence of a wildcard (*) Address may not be present in all sites.

sufoDXT?byAddress=Ecclesall

byZip=pattern select data whose postal code contains a given pattern. Pattern is case sensitive, postal code information may not be present in all sites.

sufoDXT?byZip=S1

5.5 Data content options

5.5.1 Adding columns

The first set of options allow users to add columns to the output tables

addgeoloc=1 show geolocation as individual columns.

sufoDXT?addgeoloc=1

addpairid=1 show the identifier known as **pairID**, a composite of the **siteID** and the **sensorID**.

sufoDXT?addpairid=1

addtime_full=1 Adds columns for year, month, day, day of the week, day of the year, hours, minutes, and seconds.

sufoDXT?addtime_full=1

addtime_ymd=1 Add columns for year, month and day.

sufoDXT?addtime_ymd=1

addtime_hms=1 Add columns for hour, minute and secold

sufoDXT?addtime_hms=1

addtime_date=1 adds a column with the date of the observation as `YYYY-MM-DD`

sufoDXT?addtime_date=1

addtime_tod=1 Adds a column showing the time of the day as HHMMSS. Values vary from 0 to 235959.

sufoDXT?addtime_tod=1

addtime_dow=1 Adds a column showing the day of the week (DOW) as a number: 0 = Monday, 6 = Saturday

sufoDXT?addtime_dow=1

addtime_dowh=1 Adds a column showing the day of the week (DOW) in two letter English form.

sufoDXT?addtime_dowh=1

addtime_doy=1 Adds a column showing the day of the year (1 - 365/6)

sufoDXT?addtime_doy=1

addtime_gmt=1 Add a column for a timeStamp in GMT (or Universal Time) Notation is ISO 8601, but it can be altered to replace the 'T' for a space, for databases like DB2.

sufoDXT?addtime_gmt=1

sufoDXT?addtime_gmt=1&hrtFormat=DB2

addtime_lt=1 Add a column for a timeStamp in Local Time (customer time, or clock's time). Local Time is identical to Universal Time during the winter, but it goes to BST in the summer. Notation is ISO 8601, but it can be altered to replace the 'T' for a space, for databases like DB2.

sufoDXT?addtime_lt=1

sufoDXT?addtime_lt=1&hrtFormat=DB2

addtime_msm=1 Add a column showing minutes since midnight (Local time). Values vary from 0 to 1439.

sufoDXT?addtime_msm=1

addtime_ssm=1 Add a column showing seconds since midnight (Local time). Values vary from 0 to 86399.

sufoDXT?addtime_ssm=1

addtimeinms=1 Show Unix time in milliseconds since 1970.0.

sufoDXT?addtimeinms=1

5.5.2 Modifying content arrangement

gdata Data is shown by default as one table per site/sensor combination (pair), but it is possible to group tables by sensor families (data-streams) allowing for fewer tables.

sufoDXT?gdata=byPairID this is the **default**, ie, when gdata is not specified.

sufoDXT?gdata=nogroup Explicit request not to group tables by pairID but to generate one table per data-stream.

sufoDXT?gdata=nogroup&timesort=1&addtime_LT=1 Explicit request not to group tables by pairID but to generate one table per data-stream, plus sort entries by time in each table. Local Time was requested to better appreciate the time sorting.