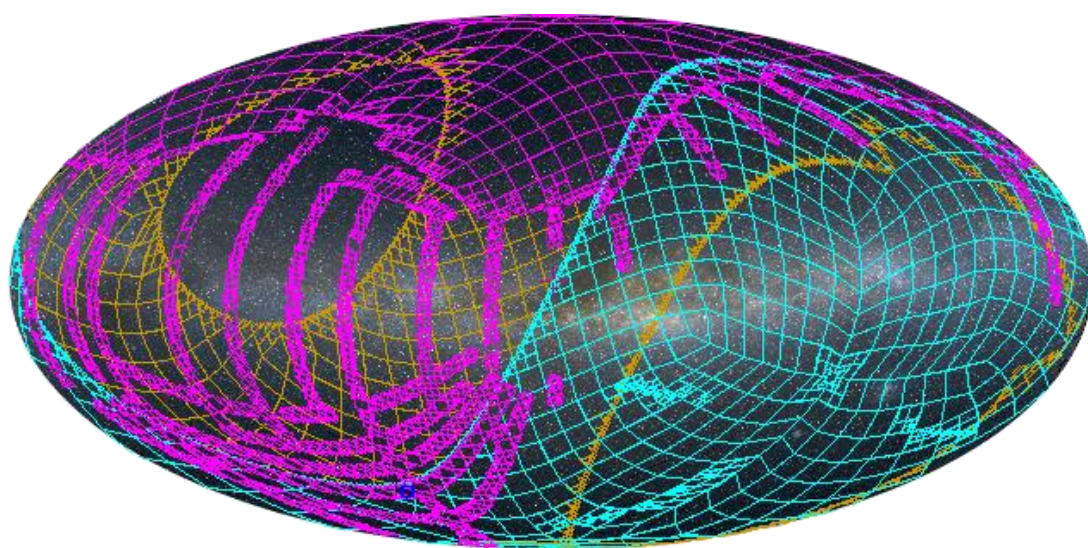


MocServer

User manual

Related to versions 5.06 and following.



Translation

with help from [deepl.com](https://www.deepl.com)

MocServer – User manual
Pierre Fernique

March 2023 (v2)

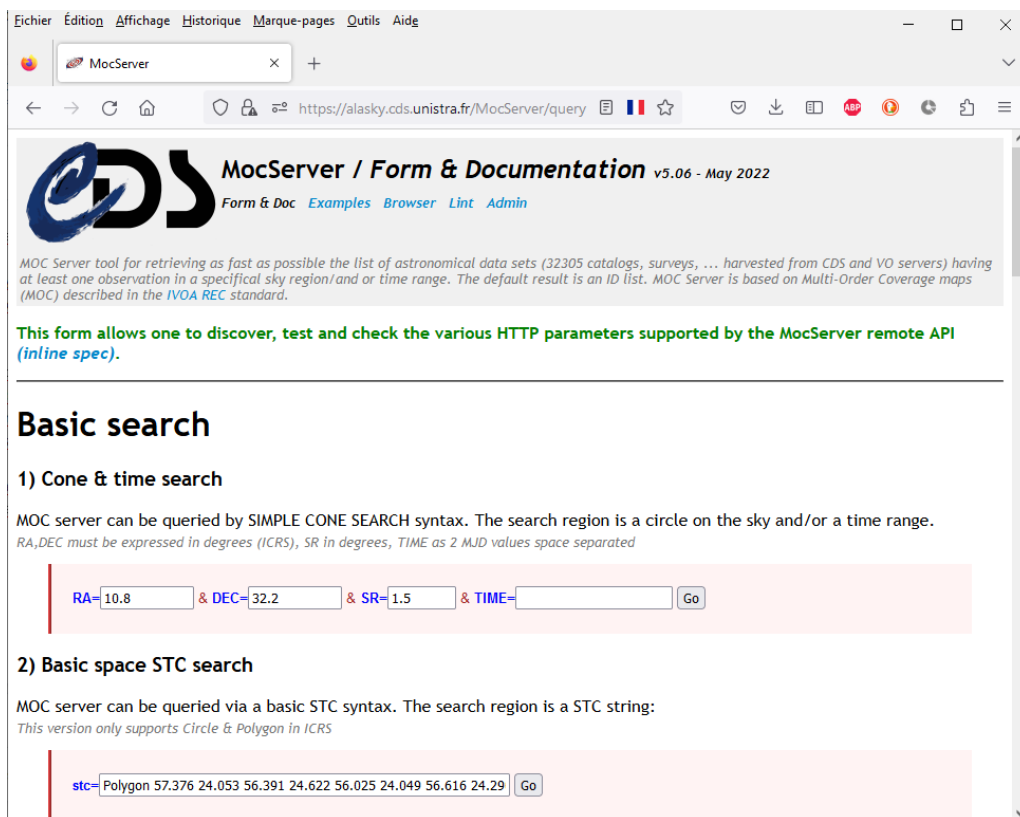
Centre de Données astronomiques de Strasbourg
Observatoire astronomique de Strasbourg

© 2023 – Université de Strasbourg / CNRS – under Open Licence (CC-BY compliant)

Introduction

The MocServer is a directory service for astronomical resources. It is based on the spatial and/or temporal coverage of the data collection. Its main goal is to provide as fast as possible (a few milliseconds) the list of astronomical resources concerned by a particular region of the sky.

The MocServer was implemented in 2015 by the Strasbourg Astronomical Data Centre (CDS). It is mainly used by CDS tools, such as Aladin Desktop, Aladin Lite, Simbad, the CDS portal, etc. In 2023 it contains 32 000 entries. This service is remotely queryable via an HTTP API. A python library is also available on Github within the "astroquery" package¹.



The screenshot shows a web browser window with the URL <https://alasky.cds.unistra.fr/MocServer/query>. The page title is "MocServer / Form & Documentation v5.06 - May 2022". The page content includes a description of the MocServer tool and a section for "Basic search". Under "1) Cone & time search", it explains the SIMPLE CONE SEARCH syntax and provides a search form with fields for RA, DEC, SR, and TIME. Under "2) Basic space STC search", it explains the basic STC syntax and provides a search form with a field for the STC string.

MocServer / Form & Documentation v5.06 - May 2022
Form & Doc Examples Browser Lint Admin

MOC Server tool for retrieving as fast as possible the list of astronomical data sets (32305 catalogs, surveys, ... harvested from CDS and VO servers) having at least one observation in a specifical sky region/and or time range. The default result is an ID list. MOC Server is based on Multi-Order Coverage maps (MOC) described in the IVOA REC standard.

This form allows one to discover, test and check the various HTTP parameters supported by the MocServer remote API (inline spec).

Basic search

1) Cone & time search

MOC server can be queried by SIMPLE CONE SEARCH syntax. The search region is a circle on the sky and/or a time range.
RA,DEC must be expressed in degrees (ICRS), SR in degrees, TIME as 2 MJD values space separated

RA=10.8 & DEC=32.2 & SR=1.5 & TIME= Go

2) Basic space STC search

MOC server can be queried via a basic STC syntax. The search region is a STC string:
This version only supports Circle & Polygon in ICRS

stc= Polygon 57.376 24.053 56.391 24.622 56.025 24.049 56.616 24.29 Go

¹ <https://astroquery.readthedocs.io/en/latest/cds/cds.html> (MocServer 4.0 compatible, only supports spatial MOCs and not temporal MOCs)

Functional principle

The MocServer manages a few thousand descriptions of astronomical data collections. Each of them is identified by a unique identifier (ID), a list of [properties](#), and most importantly the associated coverage (MOC). This coverage can be either spatial or temporal, or both. It is described using a "Multi Order Coverage map", or MOC. This is an IVOA standardised method, described in the MOC 2.0 document, which is extremely efficient in performing union, intersection and comparison operations. All of this data - the (properties, MOC) pairs for each ID - is loaded into RAM on startup of the MocServer to enable it to provide very fast responses.

A typical request to the MocServer is to ask for the list of identifiers (ID) of the collections that have at least one observation in a region of the sky. The MocServer will proceed as follows:

1. It determines the MOC corresponding to the area of the sky requested;
2. It retains the collections for which the intersection of its MOC with the area MOC is non-nil;
3. It returns the list of IDs of these collections.

In addition, it is possible to filter these results by specifying constraints on different property fields (description, provenance, copyright, etc).

The result of such a query may not be limited to identifier list, but be extended to the properties themselves, providing a "metadata directory service".

Web sites

The MocServer is accessible at the address below. It is available 24 hours a day without interruption:

<https://alasky.cds.unistra.fr/MocServer/query>

A mirror site is also available. It contains the same collection descriptions with an update time lag of up to 24 hours. It is located at the following address:

<https://alaskybis.cds.unistra.fr/MocServer/query>

These URLs without additional parameters return a web page with multiple query forms to test the different possible query parameters (see screenshot above). This is a very efficient way to test a query, and learn the syntax, without having to read this manual.

The MocServer HTML form also offers four other interfaces:

1. A list of usage examples
=> <https://.../MocServer/example>
2. A page allowing direct navigation in the managed data
=> <https://.../MocServer/browse>
3. A form for checking the compliance of a MOC with the IVOA standard
=> <https://.../MocServer/lint>
4. A remote administration page for the MocServer (restricted access)
=> <https://.../MocServer/admin>

Origins of the data

The MocServer contains several thousand descriptions of data collections. Currently they come mainly from 3 sources:

1. The HiPS lists of HiPS servers;
2. Part of the collections described in the IVOA data directory (VO registry);
3. Part of the CDS specific data collections (VizieR and Simbad).

The data is updated daily. The list of indexed collections is based on the CDS's editorial choices. It depends on the needs of the targeted client tools

Disclaimer. The MocServer is a tool to be used as a complement to the original data services. It is a directory that facilitates the selection of indexed collections, on the basis of mainly positional and temporal search criteria ("MOC"). Collection data is not ingested into the MocServer, only metadata. And the list of collections itself does not include all available metadata, or even the exhaustivity of the collections, especially for VizieR where catalogues without positional or temporal information are not included. To use the MocServer, we recommend that you check that the collections and properties you wish to work on are included. If necessary, do not hesitate to use instead or in addition the generic search mechanisms proposed by the original services, and/or an IVOA directory implementation site.

Properties of collections

The properties associated with each data collection are presented as a traditional list of "key=value" pairs grouped in a small ASCII record. These properties follow the syntax used for IVOA HiPS ².

The MocServer does not use a list of predefined keys. It will consider any property keys, even redundant ones. However, the actual use of the MocServer keys is based on the IVOA Obscore³ model and uses a great portion of its vocabulary. Thus, frequently used properties will be designated by the following keywords:

4	obs_label	= Label associated to the observations - ex: tabled
5	obs_title	= Observation title - ex: Herschel far-IR counterparts of SDSS galaxies (Dominguez+, 2014) (t..
6	obs_description	= Observation description - ex: Properties of SDSS-DR7/POP galaxies
7	obs_collection	= collection of the observations - ex: GALEX survey
8	obs_collection_label	= Label of this collection - ex: GALEX
9	obs_release_date	= observation release date - ex: 2015-05-04T12:04:45Z
10	datapoint_type	= Type of data - ex: catalog
11	bib_reference	= Bibliographic reference - ex: 2014MNRAS.441....2D
12	bib_reference_url	= URL to the bibliographic reference - ex: https://ui.adsabs.harvard.edu/?#abs/2012A%26A...544A.156M
13	obs_ack	= Acknowledgement sentence - ex: Data products from observations made with ESO Telescopes at the La .
14	obs_copyright	= Copy right mention - ex: IRAP/CADE
15	obs_copyright_url	= Url to copyright mention - ex: https://cdsarc.u-strasbg.fr/viz-bin/cat/J/MNRAS/441/2
16	t_min	= Start time of the observations (in MJD) - ex: 54965
17	t_max	= End time of the observations (in MJD) - ex: 56411
18	em_min	= Start wave length of the observations (in meters) - ex: 1E-5
19	em_max	= End wave length of the observations (in meters) - ex: 1E-4
20	obs_regime	= Regime - ex: Infrared

² <https://www.ivoa.net/documents/HiPS/>

³ <https://www.ivoa.net/documents/ObsCore/>

The exhaustive list of keywords managed by the MocServer, the number of their occurrences, and an example of the associated value for each of them can be obtained by mentioning the "[get=example](#)" parameter as a suffix to the MocServer URL.

obs_description	= (3216x)	ex: Properties of SUSS-UR//POP galaxies
nb_rows	= (28406x)	ex: 123
obs_regime	= (26813x)	ex: Infrared
obs_astronomy_kw	= (27278x)	ex: Galaxies
vizier_popularity	= (28405x)	ex: 0
data_ucd	= (28237x)	ex: src.redshift
cs_service_url	= (30152x)	ex: http://vizier.u-strasbg.fr/viz-bin/votable/-A?-source=J%2FMNRAS%2F441...
web_access_url	= (28405x)	ex: http://vizier.u-strasbg.fr/viz-bin/VizieR-2?-source=J%2FMNRAS%2F441...
moc_access_url	= (28536x)	ex: http://alasky.unistra.fr/footprints/tables/vizier/J_MNRAS_441_2_tab...
bib_reference	= (29731x)	ex: 2014MNRAS.441...2D
obs_description_url	= (31099x)	ex: https://cdsarc.u-strasbg.fr/viz-bin/cat/J/MNRAS/441/2
obs_copyright_url	= (29275x [+1:6])	ex: https://cdsarc.u-strasbg.fr/viz-bin/cat/J/MNRAS/441/2
dataprodukt_type	= (29602x)	ex: catalog
obs_release_date	= (28406x)	ex: 2015-05-04T12:04:45Z
obs_label	= (28405x)	ex: tabled
moc_type	= (28634x)	ex: smoc
moc_sky_fraction	= (26132x)	ex: 2.284E-6
moc_order	= (26119x)	ex: 11
obs_initial_ra	= (26119x)	ex: 163.35572842998585
obs_initial_dec	= (26119x)	ex: 57.256701591949735
nhc_initial_fov	= (26119x)	ex: 0.070679053431811713

<https://alasky.cds.unistra.fr/MocServer/query?get=example>

Identification of collections

Each data collection is designated by a unique identifier generated by the MocServer at the time of data ingestion. The syntax of this identifier follows the IVOA recommendations described in the IVOA Identifier 1.12 document (except for the prefix "ivo://"). This identifier is created according to the following rule⁴:

1. If a "[creator_did](#)" field is mentioned in the properties it will be used as the MocServer identifier;
2. Otherwise, if a "[publisher_did](#)" field is mentioned, it will be considered as an identifier;
3. Otherwise, the MocServer will use the fields "[publisher_id](#)" and "[obs_id](#)" and concatenate the two values to generate a unique identifier;
4. Otherwise, the MocServer will use the properties file name where the '/' separator character is replaced by a '_', and the filename extension ignored (e.g.: *CDS_P_DSScolor.prop* => *CDS/P/DSScolor*).

1 creator_did	= Full identifier of the observations (publisher+observation) - ex: ivo://ov-gso/P/HeIga/100
2 publisher_did	= Publisher identifier - ex: ivo://CDS
3 obs_id	= Observation identifier - ex: J/MNRAS/441/2/tabled

The identifier of a collection is always added to the collection properties under the key "[ID](#)" (in upper case).

E.g.: [ID=CDS/P/DSScolor](#)

Basic queries

The MocServer supports several query modes. The simplest mode follows the specifications of the IVOA standard "Simple Cone Search"⁵ and its temporal extension⁶. It uses part of the syntax described in these documents.

⁴ These cascading rules are linked to the difficulties of convergence of the IVOA. They cover all cases.

⁵ <https://www.ivoa.net/documents/Cover/ConeSearch-20080222.html>

⁶ <https://www.ivoa.net/documents/ConeSearch/20200828/index.html>

Query by cone

A basic method of querying the MocServer is to describe a circle on the celestial sphere by using the parameters "**RA=ddd**", "**DEC=ddd**" and "**SR=ddd**". RA and DEC express in decimal degrees the centre of the circle in the ICRS equatorial reference system, and SR indicates in degrees the radius of the circle.

E.g.: **RA=10.8 DEC=32.2 SR=1.5**

These parameters are used as a suffix to the MocServer url, respecting the classic http encoding rules ('?' before the parameters, '&' as a separator, some characters that must be http encoded, notably the space in "%2B").

E.g.: <https://alaska.cds.unistra.fr/MocServer/query?RA=10.8&DEC=32.2&SR=1.5>

This URL can be used directly in a web browser, or through a tool such as wget or curl. The default result will be a list of identifiers of collections having at least one observation in this search cone. The response to such a query will take only a few milliseconds whatever the size of the search circle⁷.

```
$ curl "https://alaska.cds.unistra.fr/MocServer/query?RA=10.8&DEC=32.2&SR=1.5"
CDS/B/assocdata/obscore
CDS/B/cb/lmxbdata
CDS/B/cfht/cfht
CDS/B/cfht/obscore
CDS/B/chandra/chandra
CDS/B/dao/obscore
CDS/B/eso/eso_arc
CDS/B/gcvs/gcvs_cat
CDS/B/gcvs/nsv_cat
CDS/B/gemini/obscore
```

Query by polygon

As an alternative to the cone, the MocServer can interpret a polygon query in the form of an STC-S expression, derived from the IVOA "Simple Time Coordinate" standard⁸. The parameter "**stc=xxx**" designates a polygon on the celestial sphere using a list of "right ascension" and "declination" pairs expressed in decimal degrees in the ICRS reference.

E.g.: **stc=Polygon 57.376 24.053 56.391 24.622 56.025 24.049 56.616 24.290**

Note that the MocServer only supports the basic STC-S spatial expressions *Polygon* and *Circle*, without considering union or intersection operations.

Query by time interval

When the query does not concern a region of the sky, but a time interval, the parameter to be used will be "**TIME=mjd1 mjd2**". *mjd1* is the starting date, and *mjd2* is the ending date, both expressed in "Modified Julian Date". To convert a classic date into MJD format, there are many converters available on the web⁹.

E.g.: **TIME=39619.44097 59977.7875**

Thus, the above query will return all collections with at least one observation between May 9, 1967 at 10:35 and February 2, 2023 at 18:54.

⁷ In the following, for brevity, the URL prefix will be omitted ("http ... ?"), only the parameters will be mentioned.

⁸ <http://www.ivoa.net/Documents/latest/STC-S.html>

⁹ NASA-HEASARC converter: <https://heasarc.gsfc.nasa.gov/cgi-bin/Tools/xTime/xTime.pl>

```
$ curl "https://alasky.cds.unistra.fr/MocServer/query?TIME=39619.44097+59977.7875"
CDS/B/astorb/astorb
CDS/B/comets/comets
CDS/C/CALIFA/V500/DR2
CDS/C/CGPS-HI
CDS/C/GALFAHI/Narrow
CDS/C/GALFAHI/Narrow/DR2
CDS/C/GALFAHI/Wide/DR2
```

Spatial and temporal constraints

The spatial and temporal search criteria can be combined. Only collections that simultaneously meet both criteria will be selected.

The "**intersect=xxx**" parameter allows you to specify whether the spatio-temporal area of the query should be: "**enclosed**" - completely included in the coverage of the collections to be retained, "**covers**" - completely covering it, or "**overlaps**" - simply with a non-null intersection (the default mode).

When the MocServer does not know the spatial (resp. temporal) coverage of a collection, it is not selected.

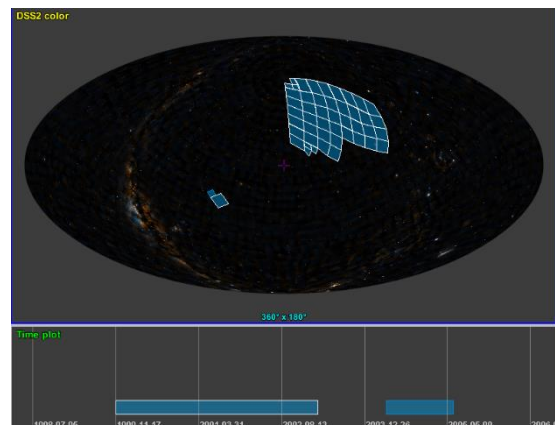
Query by MOC

As an alternative to cone, polygon and time interval queries, the MocServer also supports direct MOC queries. With this method it is possible to describe very precisely the spatio-temporal criteria of the query. The description of this query MOC can be done "in line", or via a "url" or even "uploaded".

Inline MOC: The "**moc=xxx**" parameter is used to describe an ASCII MOC according to the IVOA MOC 2 standard¹⁰.

Thus, the example below will query the MocServer in two disjoint spatial regions, each associated with a specific time interval (see above).

E.g.: **moc=** t16/6020-6022 s3/128 4/345-510 t16/6024 s4/514-516



MOC via URL: To use serialized MOCs in a file, the "**url=xxx**" parameter is required to specify the web address of the MOC that will be used to query the MocServer. The MocServer supports the MOC ASCII, FITS and even JSON representation (see IVOA MOC 2.0 document and its appendix).

E.g.: **url=**http://alasky.u-strasbg.fr/SDSS/DR9/color/Moc.fits

Uploaded MOC: The MocServer can also handle an uploaded MOC via a POST multipart "**moc**" request with the following HTML parameters:

```
<FORM enctype="multipart/form-data" method="post">
  <input name="moc" type="file"/>
  ...
</FORM>
```

¹⁰ <https://ivoa.net/documents/MOC/>

5) File MOC search

MOC server can be queried by a local MOC file. The search coverage is a local MOC uploaded via Multipart POST method (multipartID: moc).
MOC must be a local file (FITS, ASCII or JSON syntax)

Parcourir... Aucun fichier sélectionné. Go

The generation and manipulation of spatial, temporal or spatio-temporal MOCs are based on several libraries available in python¹¹, java¹², ... as well as on software, notably Aladin Desktop¹³.

Alternative spatial reference systems: Celestial MOCs are always expressed in the ICRS equatorial reference frame. However, the MocServer also manages collections associated with planets, and for each of them, the MOC is expressed in the planet's reference frame. The "[spacesys=xxx](#)" parameter is used to indicate the spatial reference frame (*equatorial, mercury, earth, moon*¹⁴, etc.) of the MOC passed in parameter. Collections that do not belong to the same spatial reference frame will not be considered¹⁵.

Query by properties

The MocServer query can also support common selection criteria based on the properties associated with each data collection. These filterings can be specified with or without associated spatial and/or temporal constraints.

Note that the values specified in the filters below are case sensitive. This can be overridden by using the "[casesensitive=false](#)" parameter.

Basic method with parameters added to the URL

Adding a constraint on a property can be done by attaching the key of the field and the constraint on that field directly to the request URL as an HTTP parameter.

Wildcards can be used: '*' - any string of characters, even empty, '?' - any character. The example below selects all collections with an "*obs_title*" property with a value containing the word "NASA".

E.g.: [obs_title=*NASA*](#)

This constraint will be appended to the MocServer query URL. If other spatial or temporal constraints are specified, the selected collections will have to satisfy all conditions.

E.g.: [https://.../query?RA=10.8&DEC=32.2&SR=1.5&obs_title=*NASA*](#)

¹¹ <https://github.com/cds-astro/mocpy>

¹² <https://wiki.ivoa.net/twiki/bin/view/IVOA/MocInfo>

¹³ <https://aladin.cds.unistra.fr/AladinDesktop/>

¹⁴ The syntax for describing planetary space frames is not yet defined by the IVOA. The MocServer currently uses the name of the planet/body in English and in lower case.

¹⁵ Note that the "[timesys=xxx](#)" parameter is already implemented. It will allow to specify the MOC time reference frame passed as a parameter. At present, the MocServer only contains data whose time information is expressed in the TCB time system (Barycentric Coordinate Time).

Wildcards on keywords: The wildcards '*' and '?' can also be used to designate a set of keys. Thus the example below will search for the word "ESO" in the set of fields beginning with "obs_".

E.g.: `obs_*=*ESO*`

Multiple values: The use of the ',' (comma) separator allows you to indicate multiple filter alternatives for the same field. The example below will search the "obs_regime" field for the value "Infrared" or "Radio".

E.g.: `obs_regime=Infrared, Radio`

Multiple keys: In the same way as for multiple values, the ',' (comma) separator allows you to indicate several keys concerned by the filter, with or without the use of wildcards ('*' and '?'). At least one of the fields concerned must meet the condition to select the collection (logical OR implicit).

E.g.: `obs_title, obs_collection=*ESAC*`

Multiple constraints: When several constraints have to be validated simultaneously, it will be necessary to indicate as many "key=value" parameters as there are constraints (implicit AND logic). The example below, unlike the previous example, will select collections where the word "ESAC" is present in both the "obs_title" field and the "obs_collection" field.

E.g.: `obs_title=*ESAC* obs_collection=*ESAC*`

Complementary result: The '!' character (exclamation mark), as a prefix to the value, will indicate the complementary result. This is equivalent to select the collections where the specified field does not contain the searched string. **Note that the '!' character is placed as a prefix to the value**, and not before the '=' character.

E.g.: `obs_id!=*CDS*`

Tip : The selection of collections whose properties do not contain a particular field will be possible by using the "`cle= !*`" filter..

The use of the complementary result on a filtering specifying multiple keys (by list or by wildcards) is not recommended as it is easily subject to misinterpretation of the implicit boolean logic. It is preferable to use the "explicit algebraic expression" method described in the following section.

Numerical comparisons: When the values are numbers, it is possible to use the comparison functions: '>' - greater than, '<' - less than, '>=' - greater than or equal to, and '<=' - less than or equal to, or a range of values via the separator ".." (two colons). In the case of comparison functions, it will be provided as a prefix to the value, and not as a replacement for the '=' character (specific to the syntax of URL parameters). Non-numeric values will never be retained.

E.g.1: `em_min=>10`

E.g.2: `t_min=52000..54000`

Calendar comparisons: As with numbers, the fields used for dates can also be compared. Supported date formats are ISO, with or without indication of hours and minutes (`yyyy-mm-dd` and `yyyy-mm-ddThh:mm:ss`), usual dates (`dd/mm/yyyy`), or the year (`yyyy`). Non-calendar values will not be considered.

Disambiguity: If the filter on a property uses a keyword that is also one of the parameters of the MocServer, namely *RA*, *DEC*, *SR*, *TIME*, *stc*, *moc*, *url*, *spacesys*, *timesys*, *fmt*, *get*, *fields*, *casesensitive*, *intersect*, *MAXREC*, *expr*, *op*¹⁶, the keyword must be preceded by the prefix "filter".

E.g.1: `filter.fmt=*jpeg*`
E.g.2: `filter.TIME=13:* & TIME=3969 59977`

Advanced method by algebraic expression

If the filter rules become more complex, it is preferable to opt for the advanced method allowed by the "`expr=xxx`" parameter. This parameter allows you to specify a complete algebraic expression using the classic set operators: "`||`" - union, "`&&`" - intersection, "`!&`" - subtraction, and the parenthesis mechanisms. As with the basic method, the use of wildcards and lists is supported.

The following example will consider collections for which the ID begins with "xcatdb" or for which the regime (*obs_regime*) is either "X-ray" or "UV", and having a HiPS URL (*hips_service_url*), except for those collections where one of the fields beginning with "obs_" contains the word "CONSTEL".

```
expr=((ID=xcatdb* || obs_regime=X-ray,UV) && hips_service_url=*) &! obs_*=*CONSTEL*
```

Result of the query

By default, the MocServer returns the collections concerned by spatial, temporal and/or property constraints, always sorted in ascending lexicographical order of record IDs. The "`get=xxx`" parameter allows to specify the form of this result. It can be a list of identifiers, the number of collections, or the properties of the collections. These results are provided by default in ASCII format. The JSON alternative is also possible (see end of this paragraph).

List of IDs. The identifiers of the selected collections are returned in the form of a list, one identifier per line (see paragraph above). This is the default mode, equivalent to using the "`get=id`" parameter.

Number of collections. The use of the "`get=number`" parameter will restrict the result to the number of concerned collections.

E.g.: `RA=10.8 DEC=32.2 SR=1.5 get=number`

```
$ curl "https://alaska.cds.unistra.fr/MocServer/query?RA=10.8&DEC=32.2&SR=1.5&get=number"
1889:
```

Properties of collections. The parameter "`get=record`" returns, for each collection concerned, the list of its properties. The result is in the form of "`key=value`" records as described above. Each record is separated from the previous one by an empty line. The use of the "`fields=xxx`" parameter allows you to specify explicitly the desired fields in the form of a comma-separated list of keys. The use of wildcards is also possible: "`*`" - any string of characters (even empty), "`?`" - any character.

E.g.: `RA=10.8 DEC=32.2 SR=1.5 get=record fields=ID,obs_tit*`

¹⁶ Ainsi que *SIZE*, *POS* et *callback* toujours supportés pour compatibilité avec les versions précédentes du MocServer.

```
$ curl "https://alaska.cds.unistra.fr/MocServer/query?RA=10.8&DEC=32.2&SR=1.5&get=record&fields=ID,obs_tit*"
ID          = CDS/B/assocdata/obscore
obs_title   = Associated data in VizieR (G.Landais, 2016) (obscore)

ID          = CDS/B/cb/lmxbdata
obs_title   = Cataclysmic Binaries, LMXBs, and related objects (Ritter+, 2004) (lmxbdata)

ID          = CDS/B/cfht/cfht
obs_title   = Log of CFHT Exposures (CADC, 1979-) (cfht)

ID          = CDS/B/cfht/obscore
obs_title   = Log of CFHT Exposures (CADC, 1979-) (obscore)
```

Limitation of the number of responses. The "**MAXREC=nn**" parameter, in accordance with the IVOA "Simple Cone Search" standard, limits the number of expected responses.

Alternative formats: As described above, the default format of a response is ASCII. The "**fmt=json**" parameter allows you to obtain the same result but packaged in JSON.

E.g.: ... **get=record fields=ID,obs_tit* fmt=json**

```
$ curl "https://alaska.cds.unistra.fr/MocServer/query?RA=10.8&DEC=32.2&SR=1.5&get=record&fields=ID,obs_tit*&fmt=json"
[
  { "ID": "CDS/B/assocdata/obscore", "obs_title": "Associated data in VizieR (G.Landais, 2016) (obscore)" },
  { "ID": "CDS/B/cb/lmxbdata", "obs_title": "Cataclysmic Binaries, LMXBs, and related objects (Ritter+, 2004) (lmxbdata)" },
  { "ID": "CDS/B/cfht/cfht", "obs_title": "Log of CFHT Exposures (CADC, 1979-) (cfht)" },
  { "ID": "CDS/B/cfht/obscore", "obs_title": "Log of CFHT Exposures (CADC, 1979-) (obscore)" },
  { "ID": "CDS/B/chandra/chandra", "obs_title": "The Chandra Archive Log (CXC, 1999-2014) (chandra)" },
  { "ID": "CDS/B/dao/obscore", "obs_title": "DAO Science Archive observations (CADC, 2020) (obscore)" },
  { "ID": "CDS/B/eso/eso_arc", "obs_title": "ESO Science Archive Catalog (ESO, 1991-2022) (eso_arc)" },
  { "ID": "CDS/B/eso/eso_cat", "obs_title": "General Catalogue of Variable Stars (GCVS, 2007-2017) (eso_cat)" }
]
```

Warning: In the case of duplicate fields (e.g.: **obs_regime=X-ray**, **obs_regime=UV**), the value will be returned using JSON list syntax (e.g.: ["X-ray", "UV"])

Result as MOC coverage

As an alternative to the identifiers or records of the selected collections, the "**get**" parameter may be used to ask the MocServer to return the coverage of the selected collections in the form of a MOC. The most obvious example is to ask for the coverage of a specific collection designated by its identifier, which means using the MocServer as a "basic Moc server".

E.g.: ID= CDS/P/SDSS9/u **get=moc**

Generation rules: By default the MOC generated as a result of a query on the MocServer follows the rules below:

1. The resulting MOC will be the union of all the coverage of the selected collections
2. Secondly, if spatial and/or temporal constraints have been specified, the final result will be the intersection of the MOC obtained in step 1 with the MOC resulting from the spatial and/or temporal constraints.

The "**op=xxx**" parameter allows this default behaviour to be changed by explicitly specifying "**union**" - the union of the relevant MOCs, or "**intersection**" - the intersection.

MOC specialisation: The "**get=xxx**" parameter allows you to specify which type of MOC should be returned, spatial MOC (*SMOC*), temporal MOC (*TMOC*) or spatio-temporal MOC (*STMOC*). This parameter can take the following values: "**anymoc**" - a MOC where the type depends on the selected collections, "**smoc**" - only the spatial component of the resultant MOC, "**tmoc**" - only the temporal component of the resultant MOC, "**stmoc**" - a spatio-temporal MOC whatever the type of the resultant MOC. In this last case, if one of the MOCs that constitute the response has no spatial (resp. temporal) dimension, the response will be considered as full, i.e. the whole sky (resp. at any date).

For compatibility with previous versions of the MocServer, the value "[moc](#)" is a synonym for "[smoc](#)" and will only return the spatial component of the MOC result.

Alternative formats: The default format of the MOC is "[FITS](#)" - recommended by the MOC 2.0 IVOA document. This format can be changed with the parameter "[fmt=xxx](#)" where xxx can be [ascii](#) - MOC ASCII format or [json](#) - MOC JSON format.

Resolution: By default, the returned MOC uses the resolution of the MOCs stored in the MocServer. It is determined by its "*MOC order*" in the MOC terminology. It is possible to modify this definition by using the parameter "[order=xxx](#)" where xxx indicates the "MOC order(s)". These orders can be specified explicitly by a value and/or indirectly by indicating a maximum size of the resulting MOC. The use of the prefix '[s](#)', respectively '[t](#)', will differentiate spatial MOC order, and temporal MOC order. For compatibility with previous versions of the MocServer, without a prefix, the spatial MOC order will be considered. The maximum size is designated by the '[<](#)' character followed by a value and a unit (without spaces between the elements). The order will be automatically reduced until the returned MOC is smaller than the indicated limit (recognised units: [MB](#)-megabyte, [KB](#)-kilobyte).

E.g.: ID= CDS/P/SDSS9/u get=stmoc [fmt=ascii](#) [order=s3](#) [t45](#) [<10KB](#)

Data synchronization for a third party directory

The MocServer can also be used as a source for a third-party directory which needs to retrieve the list of records from the MocServer, and have an efficient way to keep it up to date (see next section).

Using the multipart POST request "[maj](#)" will upload an ASCII file containing all the IDs of the previously downloaded collections.

```
<form enctype="multipart/form-data" method="post">
  <input name="maj" type="file">
</form>
```

The MocServer will return all the records of the collections not yet listed, in the form of "*key=value\nkey=value\n...*" paragraphs, separated by an empty line. It will also design the records that have been deleted by associating to them the field "[MOCSERVER_REMOVE=true](#)".

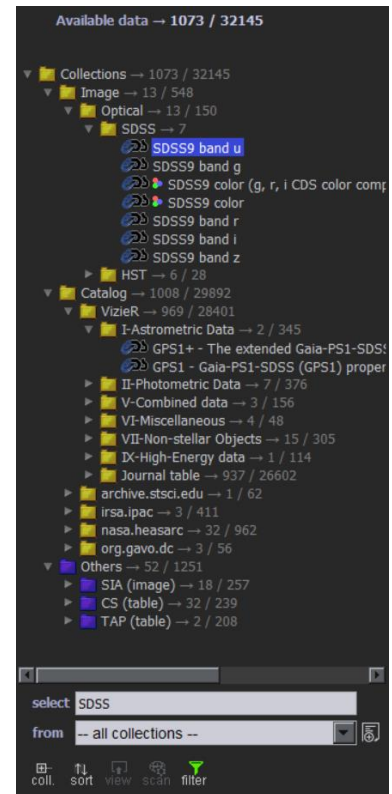
The addition of a time stamp to each of the submitted IDs (e.g.: *CDS/P/DSScolour=1664225262000*) will add to the list of new records returned those for which the date is posterior. Note that the MocServer systematically adds a "[TIMESTAMP](#)" field to the properties of the collections, providing the update date of each record (in milliseconds since 1 January 1970). It is this time stamp that the third-party application will have to reuse to ensure that the records are up to date.

Finally, if the uploaded file is reduced to a single date stamp, all posterior records will be returned.

Use of the MocServer by Aladin Desktop

Aladin Desktop software is very strongly linked to the MocServer and most of the MocServer functionalities are accessible via the Aladin graphic interface.

At each start-up, Aladin communicates with the MocServer to retrieve all the properties of the collections and builds a graphical tree allowing the user to explore the available data. When the Aladin user visualises a region of the sky, this tree is coloured to indicate in green the collections that have at least one observation in the field, and in orange those that have none. The branches that remain white reveal the collections for which the MocServer has no associated spatial MOC. In the same way, if the user visualises a temporal graph, the data tree will be coloured according to the time interval displayed.



Examples of use

The list of examples below are taken from the MocServer interface, and accessible via the following url:

<https://alasky.cds.unistra.fr/MocServer/example>

- **Around M31** - IDs of all data sets in 5 deg around M31

Returns the ID list of all data sets overlapping a circle of 10 degree diameter centered on M31

→ <http://.../MocServer/query?RA=10.67305&DEC=41.26875&SR=5>

- **In 2000** - IDs of all data sets in 2000

Returns the ID list of all data sets having on observation during the year 2000

→ <http://.../MocServer/query?TIME=51544+51910>

- **Overlapping a STC polygon** - IDs of all data sets overlapping a polygon

→ <http://.../MocServer/query?stc=Polygon+57.376+24.053+56.391+24.622+56.025+24.049+56.616+24.290>

- **Containing a STC polygon** - IDs of all data sets fully containing a polygon

Returns the ID list of all data sets which contains the area described by the STC polygon

→

<http://.../MocServer/query?stc=Polygon+312.39906+31.76699++312.66095+31.53000+312.64826+31.01805+312.45437+30.43480+312.32075+30.75116+312.28911+31.11123+312.21508+31.50335&intersect=enclosed>

- **HST data sets over SDSS** - IDs of HST collections overlapping SDSS observations

Returns the ID list of all data sets overlapping SDSS MOC (downloading by URL) and provides those for which their ID contains the substring "P/HST/"

→ http://.../MocServer/query?ID=*P/HST/*&url=http%3A%2F%2Falasky.unistra.fr%2FSDSS%2FDR9%2Fcolor%2FMoc.fits

- **Overlapping an STMOC** - IDs of all data sets overlapping an inline STMOC

Returns the ID list of all data sets overlapping the inline MOC t16/6020-6022 s3/128 4/345-510

→ <http://.../MocServer/query?moc=t16%2F6020-6022+s3%2F128+4%2F345-510>

- **SCUBA2 spatial coverage** - Spatial MOC of SCUBA2 observations

Provides the spatial MOC of the SCUBA2 850em observations. The MOC is provided in FITS format. For displaying the result, clic and drag this URL in Aladin Desktop.

→ <http://.../MocServer/query?ID=CDS/P/SCUBA2/850em&get=smoc>

- **GALEX & HERSCHEL spatial union** - Spatial MOC of the GALEX and HERSCHEL coverage

Provides the union of spatial MOCs of GALEX-NUV and HERSCHEL-PACS observations

→ <http://.../MocServer/query?ID=CDS/P/GALEXGR6/AIS/NUV,ESAVO/P/HERSCHEL/PACS-color&get=smoc>

- **A&AS spatial coverage** - Spatial MOC of all A&AS tables

Provides the union of spatial MOCs of all tables published in A&AS. The resulting MOC is provided in JSON format (not IVOA standard)

→ http://.../MocServer/query?ID=CDS/J/A%2BAS/*&get=smoc&fmt=json

- **AJ temporal coverage** - Temporal MOC of all Aj tables

Provides the union of temporal MOCs all all tables published in AJ.

→ http://.../MocServer/query?ID=CDS/J/AJ*&get=tmoc

- **SDSS & GALEX spatial intersection** - Spatial MOC of the intersection of SDSS coverage and GALEX coverage

Provides the spatio-temporal MOC of the LAMOST DR4 observations. This STMOC is provided in FITS format

→ http://.../MocServer/query?ID=*SDSS9/g.*GR6/AIS/FUV&get=smoc&op=intersection

- **Spatio-temporal coverage of LAMOST** - Spatio-temporal MOC of the LAMOST DR4 observations
Provides the spatio-temporal MOC of all coincidental observations in time and space of LAMOST & ALMA calibrator cat observations
→ <http://.../MocServer/query?ID=CDS/V/153/dr4&get=stmoc>
- **LAMOST & ALMA calibrator cat co-coincidental coverage** - Spatio-temporal MOC of the LAMOST & ALMA calibrator cat coincidental observations
Return the ID list of all resources having coincidental observations with SDSS DR9 quasar catalog
→ <http://.../MocServer/query?ID=CDS/V/153/dr4,CDS/J/MNRAS/485/1188/acccat&get=stmoc&op=intersection>
- **IDs of STMOC intersection** - IDs of STMOC intersection with SDSS DR9 quasar catalog
→ http://.../MocServer/query?ID=moc_type=stmoc&moc_time_range=>1&url=http%3A%2F%2Falasky.unistra.fr%2FMocServer%2Fquery%3FID%3DCDS%2FV%2F153%2Fdr4%2Fdr9q%26get%3Dstmoc
- **xcatdb properties** - Properties of all data sets provides by xcatdb
→ http://.../MocServer/query?ID=xcatdb*&get=record
- **CDS HiPS pixel data sets** - IDs of all CDS HiPS pixel data sets
Returns the ID list of all HiPS pixel data sets generated by CDS
→ http://.../MocServer/query?CDS/P/*
- **Radio data sets** - IDs of all Radio data sets
Returns the ID list of all data sets having an "obs_regime" or a "client_category" property containing the "Radio" word
→ http://.../MocServer/query?obs_regime,client_category=*Radio*
- **NASA related records** - Properties of all NASA related data sets
Returns the properties of all data sets having at least one field containing the "NASA" word
→ http://.../MocServer/query?*=*NASA*&casesensitive=false&get=record
- **NASA counter** - Number of records NASA related
Returns the number of records having at least one field containing the "NASA" word
→ http://.../MocServer/query?*=*NASA*&casesensitive=false&get=number
- **Seyfert location** - MOC of all Seyfert data set
Provides the union of MOCs of all data sets having an "obs_astronomy_kw" property starting with "Seyfert"
→ http://.../MocServer/query?obs_astronomy_kw=Seyfert*&get=moc
- **OIII & velocity in HST coverage** - IDs of all data sets with OIII & velocity measurements overlapping HST observations
Provides the ID list of all data sets with OIII & velocity measurements, and overlapping the MOC union of all HST observations. In this example, the MocServer is used twice: one time for providing the HST MOC, and a second time for comparing it with the other data sets.
→ http://.../MocServer/query?data_ucd=*em.line.OIII*&data_ucd=*spect.dopplerVeloc*&url=http%3A%2F%2Falasky.unistra.fr%2FMocServer%2Fquery%3Fget%3Dmoc%26ID%3D*%2FP%2FHST%2F*
- **Coverage of OIII & velocity in HST** - MOC of all data sets with OIII & velocity measurements overlapping HST observations
Same request but for retrieving resulting MOC rather than IDs.
→ http://.../MocServer/query?get=moc&data_ucd=*em.line.OIII*&data_ucd=*spect.dopplerVeloc*&url=http%3A%2F%2Falasky.unistra.fr%2FMocServer%2Fquery%3Fget%3Dmoc%26ID%3D*%2FP%2FHST%2F*
- **Extended catalogs** - Description of all tables covering more than an half of the sky
Provides some properties (ID, description, sky fraction) of all tables covering an half sky or more. The output

format is JSON.

→ http://.../MocServer/query?moc_sky_fraction=>0.5&ID=!*/P/*&fields=obs_id,obs_title,moc_sky_fraction&fmt=json

- **Copyright filtering** - Advanced record filtering based on field names with wildcards

Provides the records for which one copyright fields contains the word NASA or ESA, and a field contains the word DSS

→ http://.../MocServer/query?*copyright*=*NASA*,*ESA*&*=*DSS*&get=record

- **URL filtering** - HiPS URLs search

Provides the records of HiPS distributed simultaneously by saada and alasky http server

→ http://.../MocServer/query?hips_service_url*=http://saada.*&hips_service_url*=http://alasky.*&get=record

- **HiPS colored surveys** - HiPS colored surveys access

Provides the list of available HiPS colored surveys (ID & base URL) in JSON format

→ http://.../MocServer/query?hips_service_url=*&dataprod subtype=color&fields=id,hips_service_url&fmt=json

- **HiPS DSS colored** - DSS colored HiPS mirror sites

Provides the list of sites publishing the HiPS DSS colored

→ http://.../MocServer/query?ID=CDS/P/DSS2/color&fields=hips_service_url

- **HiPS Gaia resources** - Full set algebraica language filtering

Provides the ID list of HiPS Gaia data sets except simulations thanks to this expression: ((obs_*=*gaia* || ID=*gaia*) && hips_service_url=*) &! obs_*=*simu*

→

http://.../MocServer/query?expr=%28%28obs_*%3D*gaia*+%7C%7C+ID%3D*gaia*%29+%26%26+hips_service_url%3D*%29+%26%26+obs_*%3D*simu*%26casesensitive%3Dfalse

MocServer administration

This section is dedicated to people who are in charge of the Mocserver administration or who are interested in the technology used to implement such a service.

Technology and deployment

The Mocserver uses a classic "java servlet" technology operated by a Tomcat server. The operation of this servlet only requires the presence of a dedicated directory which contains all the data necessary, mainly the [properties](#) files and the [MOCs](#) files that it manages, as well as the [script\(s\)](#) necessary for the update of these files.

Configuration. A small optional configuration file "[mocserver.conf](#)" allows the administrator to specify the operating mode of the MocServer and in particular whether it is a "[master](#)" implementation - periodically updates the properties/MOC files by means of the above-mentioned scripts, or "[slave](#)" - simply uses the data as is. In the case of a "[master](#)" server, the update frequency and the number of backups can also be configured, as well as the maximum size authorised in memory for each MOC (see below).

```
alasky:/etc$ more mocserver.conf
# mocserver.conf => must be stored in /etc (and if not in /data/Moc)

# Harvest mode (manual|auto|slave) => default manual
# manual: harvesting process is controlled manually via Admin Web form
# auto: harvesting process is automatic and can be controlled via Admin Web form
# slave: MocServer is running as slave => use/reuse /data/Moc/multimoc.bin as it is. No Admin Web control form
HarvestMode = auto

# In case of harvesting (no slave mode)...

# Delay between two automatic harvests (in sec - default 86400) => see HarvestMode = auto
#HarvestDelay = 259200

# Maximum MB per MOC in RAM (=> decrease the MocOrder if too big - default 30)
MaxMemPerMoc = 10

# Number of Multimoc.bin backups (default 9)
#MaxBackup = 9

# Number of Pilot.html history (default 9)
#MaxHistory = 20
```

Update. The renewal of the MocServer content is performed through a unique "[harvest.bat](#)" script which must ensure the update of the properties files and the MOCs to manage. By default, it is executed once every 24 hours. It is supposed to create or copy all the files concerned into the current directory, which is a temporary "[Pilot](#)" directory. The names of these files must be derived from the ID of the collections concerned, replacing the "/" character with a "_", and using the extension "[.prop](#)" for properties and "[.fits](#)" for MOCs¹⁷.

```
-rw-r--r-- 1 tomcat7 tomcat7 48960 Oct 12 2021 CDS_B_chandra_chandra.fits
-rw-r--r-- 1 tomcat7 tomcat7 1125 Mar 9 05:22 CDS_B_chandra_chandra.prop
```

At the end of the script execution, the MocServer will compare the data it has in memory with those provided by the temporary directory. Then, if the comparison is validated, i.e. does not reveal too many data deletions, the MocServer will substitute the "[Data](#)" active data directory with this "[Pilot](#)" temporary directory and reload its memory contents from this new data set. Otherwise, the MocServer will not make the update effective, and will signal it on the administration interface described below for a verification and a manual acknowledgement.

Automatic or manual mode. The automatic update operation is carried out without interrupting the service and thus requires more RAM memory. This is why it is essential to configure the memory volume allocated to the servlet to at least double the current operation. If this is not possible, it will be necessary to inhibit the automatic update (cf. configuration file above: "[HarvestMode=manual](#)"), and to carry out manually the renewal of the data directly in the "[Data](#)" directory and to start again, also manually, the Tomcat server.

¹⁷ Note that the MOC content itself can be either FITS, ASCII or JSON.

Permanent data. In addition to the "Data" directory, the "Permanent" directory can be used to store properties and MOCs of so-called "permanent" collections. They will be systematically added to the collections from the "Data" directory and, if necessary, will replace the information for collections with the same identifier.

Overloads. To adjust the content without having to modify the collected properties files, the MocServer also uses the "exceptions.prop" file. This file provides the specification of "overloads" to be performed during the re-initialization of the MocServer to add, delete or modify the content of one or more fields from one or several properties files. It relies on a mechanism of rewriting rules using the "key=value" syntax of the properties files to select the records concerned (see related syntax in the "filtering by properties" section) and make the appropriate adjustments using lines prefixed by the ">" character, with the possibility of using the '+' or '-' character as a prefix to add or delete a single word.

The 2 paragraphs above will add the field "obs_regime = Gamma-ray" to all records that have a "client_category" field with a value starting with "Image/Gamma", and remove the word "AladinDesktop" from any existing "client_application" fields for records with an "ID" starting with "ESAVO":

```
client_category = Image/Gamma*
>obs_regime = Gamma-ray

ID = ESAVO*
>client_application = -AladinDesktop
```

The use of this "overload" file is very effective in practice to correct or complete properties temporarily, but it must be used with caution and "parsimony" as it is easy to "forget" that an overload rule is active, or even to define a rewrite rule that will apply later on to unwanted records. As far as possible, it is always preferable to make corrections before this last step.

Backups and logs. During an update, the MocServer keeps the previous memory content in the form of a single binary file "Multimoc.bin" generated in the "Cache" directory. This backup file and the 9 previous versions can be reused via the administration interface to immediately revert to a previous content. The MocServer also keeps in this same directory the "log" of the script as well as the result of the comparison carried out to validate the update under the respective names "HarvestLog.html" and "Pilot.html". These, as well as the 20 previous versions, can be consulted via the administration interface.

```
Pilot comparison - 2023-03-03 23:18:56...

Current multimoc: 32637 datasets using 3GB (49MB for properties) [26195 SMoc + 2521 TMoc + 1166 STMoc]
Pilot: 32647 datasets using 49MB [0 SMoc + 0 TMoc + 0 STMoc]

0 deleted, 10 added, 1347 modified, 31290 unchanged

+ CDS/3/Ap3/919/85/table1
+ CDS/3/Ap3/919/85/table14
+ CDS/3/Ap3/919/62/table4
+ CDS/3/Ap3/919/62/table3
+ CDS/3/Ap3/919/62/table5
+ nasa.heasarc/m83xrbco
+ CDS/3/Ap3/919/99/table3
+ CDS/3/Ap3/919/62/sample
+ CDS/3/Ap3/919/85/callbr1
+ CDS/3/AJ/164/244/table2
* 1339 records with "vizier_popularity" modified (see below)
* nasa.heasarc/m83xco (get it)
  * obs_description = The authors of this table have obtained a series of deep X-ray images of the nearby (4.
    =====> The authors of this table have obtained a series of deep X-ray images of the nearby (4.
* CDS/Simbad (get it)
  * hips_release_date = 2023-03-02T03:06Z ==> 2023-03-03T03:07Z
* UNK.AUTH/PeJwST-MIRI_Imaging (get it)
  - datapoint_subtype
  + hipsgen_date_7
  + hipsgen_params_7
  * hips_estsize = 7312620 ==> 12883575
  * addendum_id = APPEND/P/1673397445, APPEND/P/1673576470
    =====> APPEND/P/1673397445, APPEND/P/1673576470, APPEND/P/1677782867
```

Administration interface

The MocServer has an administration interface, with restricted access, accessible via the following url:

<https://alasky.cds.unistra.fr/MocServer/admin>

This MocServer administration interface provides a brief description of the location of the directories and configuration files described above:

Configuration

- **Configuration file:** `/etc/mocserver.conf` ([optional, in `/etc` otherwise in `/data/Moc`] allowing to modify default MocServer behavior)
- **Data directory:** `/data/Moc/Data` (directory containing current MOCs & Props used by the MocServer as a list of individual files)
- **Pilot directory:** `/data/Moc/Pilot` (directory dedicated to MOCs & Props in preparation for the next release)
- **Permanent directory:** `/data/Moc/Permanent` (directory containing local MOCs & Props, not harvested - overriding possible duplicated records)
- **Harvest script:** `/data/Moc/harvest.bat` (script called for copying/downloading MOCs & Props in Pilot directory)
- **Rewriting rules:** `/data/Moc/exceptions.prop` (rewriting rule exceptions for modifying some prop records during MocServer (re)loading process)
- **Cache directory:** `/data/Moc/Cache` (directory containing binary files (MOC&Props in one file) & statistics)

The interface also provides the current status of the Tomcat server and the MocServer servlet (see example below), as well as weekly usage statistics:

32705 datasets using 3GB (50MB for properties) [26328 SMoc + 2566 TMoc + 1166 STMoc]

Servlet status

- Apache Tomcat/7.0.88 (Debian)
- Running servlets: 4
- Number of threads: 106
- Memory size: 9.46GB
- Free memory: 3.51GB

Finally, this interface allows you to manually perform the different steps of an update, to go back to a backup, to consult the logs and the comparisons of past updates (see previous section).

Actions:

- **status** : general status of the MocServer (size, existing backups, ...)
- **harvest** : launch the harvest process for populating the pilot directory (no change)
- **pilot** : compare the current MocServer content to the pilot files (no change)
- **validate** : replace data directory by pilot files and update MocServer content according to
- **update** : update the current MocServer content from the data directory

Exhaustive list of query parameters

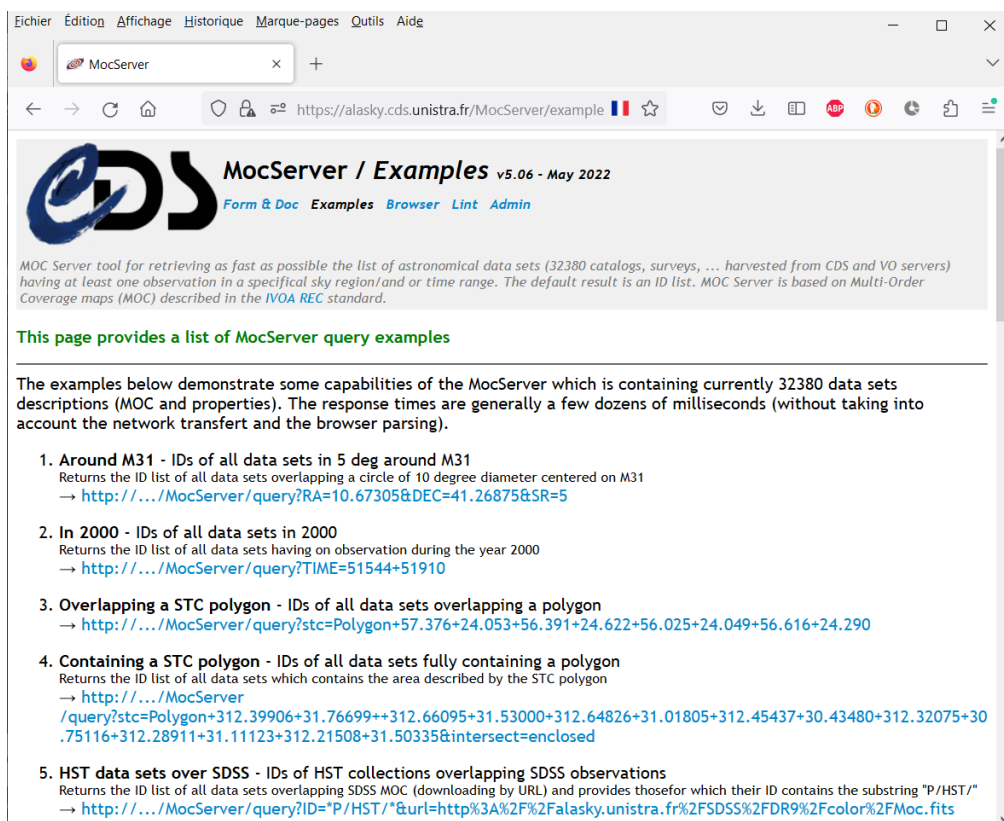
Version: v5.06 - May 2022

[Query parameters (.../query?xxx)]

help - This help. help=version => MocServer version number
RA - Input Cone Search right ascension in degrees ICRS (see IVOA CS REC)
DEC - Input Cone Search declination in degrees ICRS (see IVOA CS REC)
SR - Input Cone Search radius in degrees (see IVOA CS REC)
TIME - Input Cone Search time range as two MJD numbers (see IVOA CS REC&WD)
stc - Input STC polygon or circle region (see IVOA STC-S note)
moc - Input Inline SMOC, TMOC or STMOC (see IVOA MOC REC&WD)
spacesys - Input space system
timesys - Input time system
url - Input MOC url (FITS, ASCII or JSON syntax - see IVOA MOC REC&WD)
intersect - Input MOC intersection constraint: overlaps, enclosed, covers
expr - Input filter by expression (E.g.: obs_title=*NASA*,*CDS* && hips_service_url!=http://aladin*)
get - Output result: id, record, number, smoc, tmoc, stmoc, anymoc (deprecated: moc, umoc, imoc)
fmt - Output format: ascii, json, fits, (deprecated: asciic, glu)
order - Output MOC order limit (ex1: t32 s13, ex2: 1MB)
op - Output MOC operation: union, intersection
fields - Output record fields: (E.g.: ID,obs_title,hips_service*)
casesensitive - Output record fields case: true, false
MAXREC - Output record number
callback - [deprecated] Output callback JSON function
POS - [deprecated] => use Cone Search query
SIZE - [deprecated] => use Cone Search query

Table of contents

Introduction.....	3
Functional principle	4
Web sites	4
Origins of the data	5
Properties of collections.....	5
Identification of collections	6
Basic queries.....	6
Query by cone	7
Query by polygon	7
Query by time interval.....	7
Spatial and temporal constraints	8
Query by MOC	8
Query by properties	9
Basic method with parameters added to the URL	9
Advanced method by algebraic expression.....	11
Result of the query	11
Result as MOC coverage.....	12
Data synchronization for a third party directory.....	13
Use of the MocServer by Aladin Desktop.....	14
Examples of use.....	15
MocServer administration.....	18
Technology and deployment.....	18
Administration interface	19
Exhaustive list of query parameters.....	21



MocServer – User manual
 March 2023 version

© 2023 - Université de Strasbourg/CNRS – under Open Licence (CC-BY compatible)