

J-PLUS data access tutorial

J. Hernández & J. Varela

3/10/2017

Relational Databases

- Commodity Software in Computer Science to store data (MySQL, PostgreSQL, ORACLE, IBM DB2,...).
- Data is stored as tables where each table has a set of typed columns.
- **SQL** Structured **Q**uery **L**anguage is a common language to access relational databases.
- TAP & ADQL → framework to access Astronomical data stored in relational databases. ADQL is a subset of SQL, plus several specific functions for astronomy.

Archive Web Portal

- <http://archive.cefca.es/catalogues>
 - Several web page for 'sky navigation', image search, object search, ... (you saw it yesterday)
 - Several **V**irtual **O**bservatory services: Cone Search, SIAP & TAP
 - **TAP** (Table Access Protocol) service that executes **ADQL** queries.
 - V.O. Asynchronous Queries (ADQL) web page to launch query execution and retrieve results.

Tables in the database

Tables: Most used

- [jplus.Filter](#)
- [jplus.MagABDualObj](#)
- [jplus.MagABSingleObj](#)
- [jplus.TileImage](#)

Tables: All

- [jplus.DuplicatedFLambdaDualObj](#)
- [jplus.DuplicatedFLambdaSingleObj](#)
- [jplus.DuplicatedFNuDualObj](#)
- [jplus.DuplicatedFNuSingleObj](#)
- [jplus.DuplicatedMagABDualObj](#)
- [jplus.DuplicatedMagABSingleObj](#)
- [jplus.Filter](#)
- [jplus.FLambdaDualObj](#)
- [jplus.FLambdaSingleObj](#)
- [jplus.FNuDualObj](#)
- [jplus.FNuSingleObj](#)
- [jplus.MagABDualObj](#)
- [jplus.MagABSingleObj](#)
- [jplus.TileImage](#)
- [jplus.TileImageWithFlag](#)

Table definition

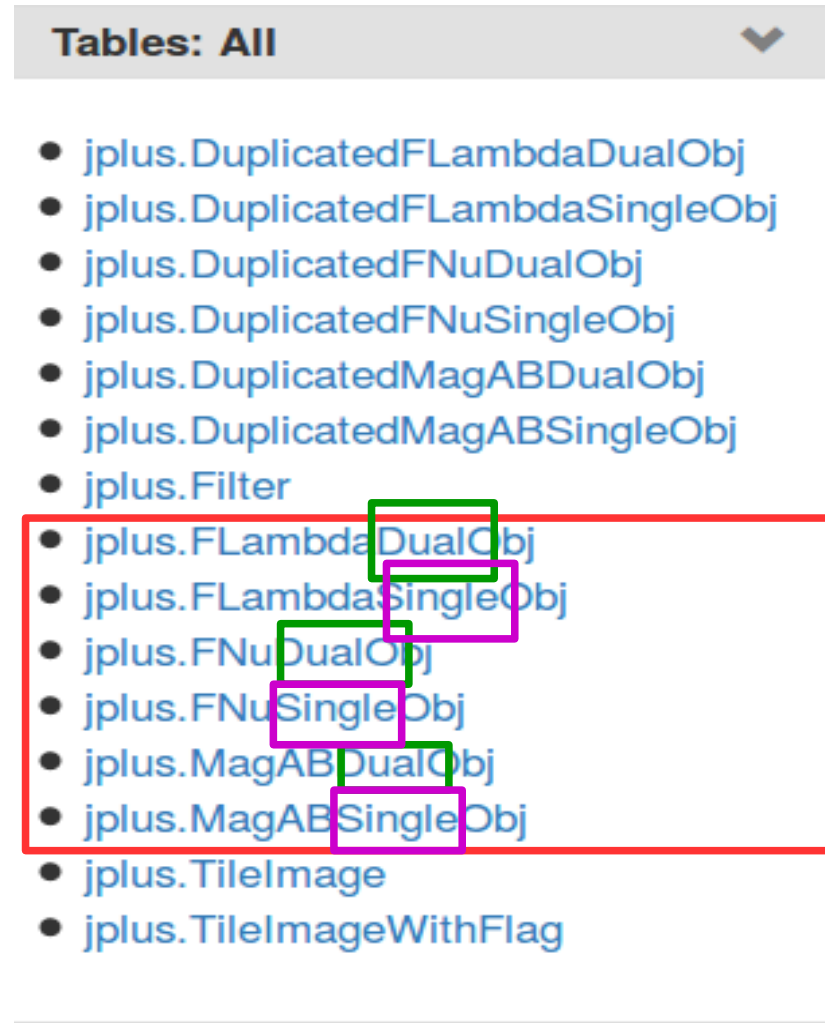
Table jplus.TileImage

Table that contains the main information and properties of the coadded images. These images have been created combining the individual processed science images for each filter. The detection and the object properties contained in the other tables have been computed from these images.

COLUMNS

Name	Type	Units	UCD	Description
TILE_ID	int		meta.id;obs.image	Identifier of the Tile image
FILTER_ID	short		instr.filter;meta.id	Identifier of the filter used for the image
REF_TILE_ID	int		meta.id	Identifier of the Tile image of reference in dual mode
name	char		meta.id;obs.image	Name of the composed tile image
RA	double	deg	pos.eq.ra;meta.main	Right ascension in equatorial coordinates
DEC	double	deg	pos.eq.dec;meta.main	Declination in equatorial coordinates
hpix5	int		pos.healpix	Healpix pixel Order 5 NSIDE 32 (NESTED schema) of image position
ZPT	float		arith.zp	Zero point of the image
ERRZPT	float		stat.error;arith.zp	Zero point error

Tables in the database



Tables in the database

- jplus.MagAB*Obj
 - AB Magnitudes
- jplus.FNu*Obj
 - Fluxes in units of 10^{-30} erg/s/cm²/Hz
- jplus.FLambda*Obj
 - Fluxes in units of 10^{-19} erg/s/cm²/Å

Tables in the database

- `jplus.Filter` → Table with information of the filters
- `jplus.TileImage` → Information of the tiles.
 - Name, coordinates, filter,...
 - `REF_TILE_ID`: `TILE_ID` of the reference image of the pointing (ie. the `TILE_ID` of the rSDSS tile)
 - Zero points (of the images)
 - FWHM statistics
 - Depth estimations from image noise
- `jplus.PhotoZ` → Table with information on photo-z's (right now empty)

ADQL

- *Astronomical Data Query Language* (ADQL) is a language used by the *International Virtual Observatory Alliance* (IVOA) to represent astronomy queries posted to VO services.
- Reference Document:
 - <http://www.ivoa.net/documents/latest/ADQL.html>
- Based on SQL Language
 - Very similar to MySQL (although some features of this could be missing)
 - Implements astronomy related features, in particular, functions dealing with distances.

V.O. Asynchronous Queries

Select queries to be shown

Refresh list

Create new query

History of queries

Help

The screenshot shows the V.O. Asynchronous Queries interface. The top navigation bar includes a 'PLUS' logo, a 'Services' dropdown, and icons for 'Refresh', 'Create Query Job', 'Historical queries', and 'ADQL help'. The left sidebar contains filters: 'VO Asynchronous Queries', 'Options', 'Show anonymous jobs' (unchecked), 'Show my jobs' (checked), and 'Show users public jobs' (unchecked), with an 'Apply' button. The main table lists query jobs with columns: Job ID, Phase, Owner, Public, Description, Quote (sec.), Available until, Start time, Exec Time, Run, and Actions. The table contains 10 rows of data, mostly with 'COMPLETED' status. Red arrows point from text labels to specific UI elements: 'Select queries to be shown' points to the filter section; 'Refresh list' points to the 'Refresh' icon; 'Create new query' points to the 'Create Query Job' icon; 'History of queries' points to the 'Historical queries' icon; 'Help' points to the 'ADQL help' icon; 'Click here to make your query public' points to the 'Public' checkbox in row 374; 'Show additional information' points to the 'Job ID' column header; 'Launch the query' points to the play icon in the 'Run' column; 'Cancel the query' points to the stop icon; 'Copy the query' points to the document icon; 'Delete/abort the query' points to the red 'X' icon; 'Download the results' points to the download icon; and 'Broadcast the results' points to the broadcast icon.

Job ID	Phase	Owner	Public	Description	Quote (sec.)	Available until	Start time	Exec Time	Run	Actions
365	COMPLETED	jvarela	☐		1	2016-11-08 10:54:27	2016-11-04 10:54:33	0 s		
366	COMPLETED	jvarela	☐		1	2016-11-08 10:56:57	2016-11-04 10:57:01	0 s		
367	COMPLETED	jvarela	☐		1	2016-11-08 11:03:41	2016-11-04 11:03:43	0 s		
374	COMPLETED	jvarela	☐		1	2016-11-09 16:45:46	2016-11-05 16:45:50	0 s		
375	ABORTED	jvarela	☐		10259	2016-11-09 16:54:47	2016-11-05 16:54:52			
376	COMPLETED	jvarela	☐		4	2016-11-09 16:59:03	2016-11-05 16:59:06	0 s		
377	COMPLETED	jvarela	☐		4	2016-11-09 17:00:17	2016-11-05 17:00:20	0 s		
378	COMPLETED	jvarela	☐		2	2016-11-09 17:06:19	2016-11-05 17:06:22	0 s		
379	COMPLETED	jvarela	☐		17	2016-11-09 17:10:47	2016-11-05 17:10:51	0 s		
380	COMPLETED	jvarela	☐		5	2016-11-09 17:23:36	2016-11-05 17:23:40	1 s		

Click here to make your query public

Show additional information

Launch the query

Cancel the query

Copy the query

Delete/abort the query

Download the results

Broadcast the results

V.O. Asynchronous Queries – New QUERY

PLUS

Services

Create new job

Query ? ADQL ? Tables: Most used ? Tables: All ? Functions ? Enumerations

ADQL query:

```
SELECT * FROM TileImage
```

Output Format: Comma separated values

Maximum # of rows: 1000000

Owner:

Description:

Note: After planning the query use the Run pending job button to launch the execution.

Validate Query Plan Query Close

ADQL

- Keywords are case-insensitive
- Strings between single quotes
- No “;” at the end of command

ADQL Commands

- **SELECT ... FROM ... WHERE ...**
 - What do we want
 - From which tables
 - Conditions that must satisfy
- **... JOIN ... ON | USING ...**
- **GROUP BY ... HAVING**
- **Functions:**
 - Math functions
 - Geometric functions

ADQL Examples

SELECT

Select all fields

```
SELECT * FROM jplus.TileImage
```

Select specific fields

```
SELECT  
    name, RA, DEC, TILE_ID  
FROM  
    jplus.TileImage
```

Use of expressions

```
SELECT  
    name, RA, DEC, TILE_ID, TEXPOSE/3600.0  
FROM  
    jplus.TileImage
```

Use of aliases

```
SELECT  
    name, RA, DEC, TILE_ID,  
    TEXPOSE/3600.0 as total_exposure_hours  
FROM  
    jplus.TileImage
```

ADQL Examples

SELECT

Select the first 100 elements (TOP)

```
SELECT TOP 100
```

```
*
```

```
FROM
```

```
  jplus.TileImage
```

Order output

```
SELECT TOP 100
```

```
*
```

```
FROM
```

```
  jplus.TileImage
```

```
ORDER BY
```

```
  ref_tile_id ASC ,filter_id DESC
```

ADQL Examples

SELECT ... WHERE

Simple WHERE

```
SELECT
  *
FROM
  jplus.TileImage
WHERE
  tile_id=2509
```

Adding "AND"

```
SELECT
  *
FROM
  jplus.TileImage
WHERE
  ref_tile_id=2509 AND calib_procedure=3
```

ADQL Examples

SELECT ... WHERE

Adding "OR"

SELECT

*

FROM

jplus.TileImage

WHERE

ref_tile_id=2509

AND

(calib_procedure=3 OR calib_procedure=4)

Using "IN"

SELECT

*

FROM

jplus.TileImage

WHERE

ref_tile_id=2509

AND

calib_procedure IN (3,4)

ADQL Examples

SELECT ... WHERE

```
# USING "NOT"
# (be aware that it looks like "!" doesn't work)
SELECT
  *
FROM
  jplus.TileImage
WHERE
  ref_tile_id=2509
  AND
  NOT calib_procedure=3

# Using "NOT IN"
SELECT
  *
FROM
  jplus.TileImage
WHERE
  ref_tile_id=2509
  AND
  calib_procedure NOT IN (1,4)
```

ADQL Examples

SELECT . . . WHERE

```
# Using BETWEEN
SELECT
    *
FROM
    jplus.TileImage
WHERE
    ra BETWEEN 100 AND 150
```

```
# Using LIKE
SELECT
    *
FROM
    jplus.TileImage
WHERE
    name LIKE '%rSDSS%'
```

ADQL Examples

Arrays and Enumerations

- Magnitudes, fluxes and FLAGS in dual-mode catalogues are arrays.
- Order of the arrays given by jplus.Filter, ie.:

rSDSS, gSDSS, iSDSS, zSDSS, uJAVA, J0378, J0395, J0410,
J0430, J0515, J0660, J0861

```
# Retrieve the coordinates and MAG_AUTO for
# objects in tile_id=2035 and CLASS_STAR<0.1.
SELECT
    ALPHA_J2000, DELTA_J2000, MAG_AUTO
FROM
    jplus.MagABDualObj
WHERE
    tile_id=2035 AND CLASS_STAR<0.1
```

ADQL Examples

Arrays and Enumerations

```
# Accesing elements of the array by enumeration.  
# Retrieve the coordinates and the (g-r) color  
# using MAG_AUTO for objects in tile_id=2035 and  
# CLASS_STAR<0.1.  
SELECT  
    ALPHA_J2000, DELTA_J2000,  
    MAG_AUTO[jplus::gSDSS] - MAG_AUTO[jplus::rSDSS] as g_r  
FROM  
    jplus.MagABDualObj  
WHERE  
    tile_id=2035 AND CLASS_STAR<0.1
```

ADQL Examples

Arrays and Enumerations

```
# Accesing elements of the array by index.  
# Retrieve the coordinates and the (g-r) color  
# using MAG_AUTO for objects in tile_id=2035 and  
# CLASS_STAR<0.1.  
SELECT  
    ALPHA_J2000, DELTA_J2000,  
    MAG_AUTO[2] - MAG_AUTO[1] as g_r  
FROM  
    jplus.MagABDualObj  
WHERE  
    tile_id=2035 AND CLASS_STAR<0.1
```

ADQL Examples

JOIN . . . ON/USING

```
SELECT
    t1.field1, . . . , t2.field1, . . . , t3.field1
FROM
    table1 AS t1
JOIN
    table2 AS t2
ON
    t1.fieldA=t2.fieldB
JOIN
    table3 AS t3
ON
    t2.fieldN=t3.fieldM
. . .
WHERE
. . .
```

ADQL Examples

JOIN . . . USING

```
# If fieldA is common between t1 & t2
SELECT
    t1.field1, . . . , t2.field1
FROM
    table1 AS t1
JOIN
    table2 AS t2
USING
    (fieldA)
. . .
WHERE
. . .
```

ADQL Examples

JOIN ... ON

EXERCISE

Adding the filter name to a query on the

single-mode catalogue

SELECT TOP 10

 t1.NUMBER, t1.ALPHA_J2000 as RA,

 t1.DELTA_J2000 as DEC,

 t2.name as filter

FROM

 jplus.MagABSingleObj t1

JOIN

 jplus.Filter t2

ON

 t1.filter_id=t2.filter_id

ADQL Examples

JOIN ... ON

EXERCISE

Computing the ratio between the FWHM of
each good stellar object and the average of the
image.

SELECT TOP 10

 t1.NUMBER,t1.X_IMAGE as x,
 t1.Y_IMAGE as y,t2.name as filter,
 (t1.FWHM_WORLD*3600)/t3.FWHMG as fwhm_ratio

FROM

 jplus.MagABSingleObj t1

JOIN

 jplus.Filter t2

ON

 t1.filter_id=t2.filter_id

JOIN

 jplus.TileImage t3

ON

 t1.tile_id=t3.tile_id

WHERE

 CLASS_STAR>0.9 AND FLAGS=0

ADQL Examples

JOIN . . . USING

```
# EXERCISE
# Adding the filter name to a query on the
# single-mode catalogue
SELECT TOP 10
  t1.NUMBER,t1.ALPHA_J2000 as RA,
  t1.DELTA_J2000 as DEC,t2.name as filter
FROM
  jplus.MagABSingleObj t1
JOIN
  jplus.Filter t2
USING
  (filter_id)
```

ADQL Examples

JOIN ... USING

```
# EXERCISE
# Computing the ratio between the FWHM of
# each good stellar object and the average of the
# image.
SELECT
  TOP 10 t1.NUMBER,t1.X_IMAGE as x,
  t1.Y_IMAGE as y,t2.name as filter,
  (t1.FWHM_WORLD*3600)/t3.FWHMG as fwhm_ratio
FROM
  jplus.MagABSingleObj t1
JOIN
  jplus.Filter t2
USING
  (filter_id)
JOIN
  jplus.TileImage t3
USING
  (tile_id)
WHERE
  CLASS_STAR>0.9 AND FLAGS=0
```

ADQL Examples

GROUP BY ... HAVING

```
SELECT
    t1.field1,...,t2.field1,...,t3.field1
FROM
    table1 AS t1
JOIN
    table2 AS t2
ON
    t1.fieldA=t2.fieldB
JOIN table3 AS t3
ON
    t2.fieldN=t3.fieldM
...
WHERE
    [conditions]
GROUP BY
    tN1.fieldM1, tN2.fieldM2, ..., tNi.fieldMi
HAVING
    [conditions on grouping fields]
```

ADQL Examples

GROUP BY ... HAVING

```
# EXERCISE
# Counting the detections on different bands
# in a given area of the sky.
SELECT
    name as filter,COUNT(*) as nobjects
FROM
    jplus.MagABSingleObj t1
JOIN
    jplus.Filter t2
USING
    (filter_id)
WHERE
    t1.ALPHA_J2000 BETWEEN 11 AND 120 AND
    t1.DELTA_J2000 BETWEEN 39 AND 40
GROUP BY
    t2.name
```

ADQL Examples

GROUP BY ... HAVING

```
# EXERCISE
# Estimating the depth of the single catalogues
# as the average MAG_AUTO for objects with
# 2.9<SNR<3.1 in a 2" aperture and selecting
# those with depth >19.
SELECT
    name, COUNT(*) as nobjects,
    AVG(1.086/MAG_ERR_APER_2_0) as snr,
    AVG(MAG_AUTO) as avg_mag
FROM
    jplus.MagABSingleObj t1
JOIN
    jplus.TileImage t2
USING
    (tile_id)
WHERE
    t1.MAG_ERR_APER_2_0>0
    AND
    (1.086/t1.MAG_ERR_APER_2_0) BETWEEN 2.9 AND 3.1
GROUP BY
    t2.name
HAVING
    AVG(MAG_AUTO)>19
```

ADQL Examples

Common Functions

- Typical mathematical functions:
 - Trigonometric ones:
 - COS, SIN, TAN, ATAN, ASIN, ACOS, ...
 - DEGREES, RADIANS
 - Powers and logarithms:
 - $\text{POWER}(X, Y) = \text{POW}(X, Y) = X^Y$
 - EXP
 - SQRT
 - LOG, LOG10
 - Rounding and truncation:
 - CEIL, FLOOR, ROUND, TRUNCATE, MOD, DIV
 - Random numbers:
 - RAND
- String functions:
 - CONCAT, LIKE, ...

ADQL Examples

ADQL Functions - Regions

- Preliminary remarks:
 - `'coord.system' ∈ ['ICRS', 'FK4', 'FK5', 'GALACTIC']`
 - `if 'coord.system' ∈ ['ICRS', 'FK4', 'FK5']`
 - **$(\alpha, \delta) = (\text{RA}, \text{DEC})$**
 - `if 'coord.system' = 'GALACTIC'`
 - **$(\alpha, \delta) = (l, b)$**
 - All units are in **degrees**.

ADQL Examples

ADQL Functions - Regions

- Functions to define “regions”:
 - **POINT** (' coord.sys ' , α , δ)
 - **CIRCLE** (' coord.sys ' , α_{centre} , δ_{centre} , radius)
 - **BOX** (' coord.sys ' , α_{centre} , δ_{centre} , width , height)
 - **POLYGON** (' coord.sys ' , α_1 , δ_1 , α_2 , δ_2 , α_3 , δ_3 , . . . , α_n , δ_n)
- Functions using “regions”:
 - **DISTANCE** (POINT1 , POINT2) $\rightarrow$ **ARC_DISTANCE** (α_1 , δ_1 , α_2 , δ_2)
 - **CONTAINS** (REGION1 , REGION2)
 - **INTERSECTS** (REGION1 , REGION2)
 - **AREA** (REGION)

ADQL Examples

ADQL Functions - Regions

```
# EXERCISE
# Computing the distance of 1000 objects in
# the dual-mode catalogue to the galactic centre.
SELECT TOP 1000
    DISTANCE (
        POINT ('ICRS', ALPHA_J2000, DELTA_J2000) ,
        POINT ('GALACTIC', 0, 0)
    )
    AS galactocentric_distance
FROM
jplus.CalibratedMagABDualObj
```

ADQL Examples

ADQL Functions - Regions

```
# EXERCISE
# Computing the number of neighbours within 1' for
# 100 objects of a given tile, and the maximum distance
# of those neighbours, sorting at the end by RA.
# WARNING: we use MAX() due to a problem with the syntax validator.
# Using DISTANCE
SELECT TOP 100 t1.number as id,
    t1.ALPHA_J2000 as RA, t1.DELTA_J2000 as DEC,
    COUNT(*) as n_neighbours,
    MAX(DISTANCE(POINT('ICRS',t1.ALPHA_J2000,t1.DELTA_J2000),
        POINT('ICRS',t2.ALPHA_J2000,t2.DELTA_J2000))) * 3600
        as max_distance_arcsec
FROM
    jplus.MagABDualObj t1,
    jplus.MagABDualObj t2
WHERE
    t1.tile_id=2035 AND t2.tile_id=2035 AND
    DISTANCE(POINT('ICRS',t1.ALPHA_J2000,t1.DELTA_J2000),
        POINT('ICRS',t2.ALPHA_J2000,t2.DELTA_J2000)) < 1/60.
GROUP BY
    t1.number,t1.ALPHA_J2000,t1.DELTA_J2000
ORDER BY
    RA
```

ADQL Examples

ADQL Functions - Regions

```
# EXERCISE
# Computing the number of neighbours within 1' for
# 100 objects of a given tile, and the maximum distance
# of those neighbours, sorting at the end by RA.
# WARNING: we use MAX() due to a problem with the syntax validator.
# Using CONTAINS
SELECT TOP 100 (t1.number) as id,
    t1.ALPHA_J2000 as RA, t1.DELTA_J2000 as DEC,
    COUNT(*) as n_neighbours,
    MAX(DISTANCE(POINT('ICRS',t1.ALPHA_J2000,t1.DELTA_J2000),
        POINT('ICRS',t2.ALPHA_J2000,t2.DELTA_J2000))) * 3600
        as max_distance_arcsec
FROM
    jplus.MagABDualObj t1,
    jplus.MagABDualObj t2
WHERE
    t1.tile_id=2035 AND t2.tile_id=2035 AND
    1=CONTAINS(POINT('ICRS',t2.ALPHA_J2000,t2.DELTA_J2000),
        CIRCLE('ICRS',t1.ALPHA_J2000,t1.DELTA_J2000,1/60.))
GROUP BY
    t1.number,t1.ALPHA_J2000,t1.DELTA_J2000
ORDER BY
    RA
```

ADQL Examples

ADQL Functions - Regions

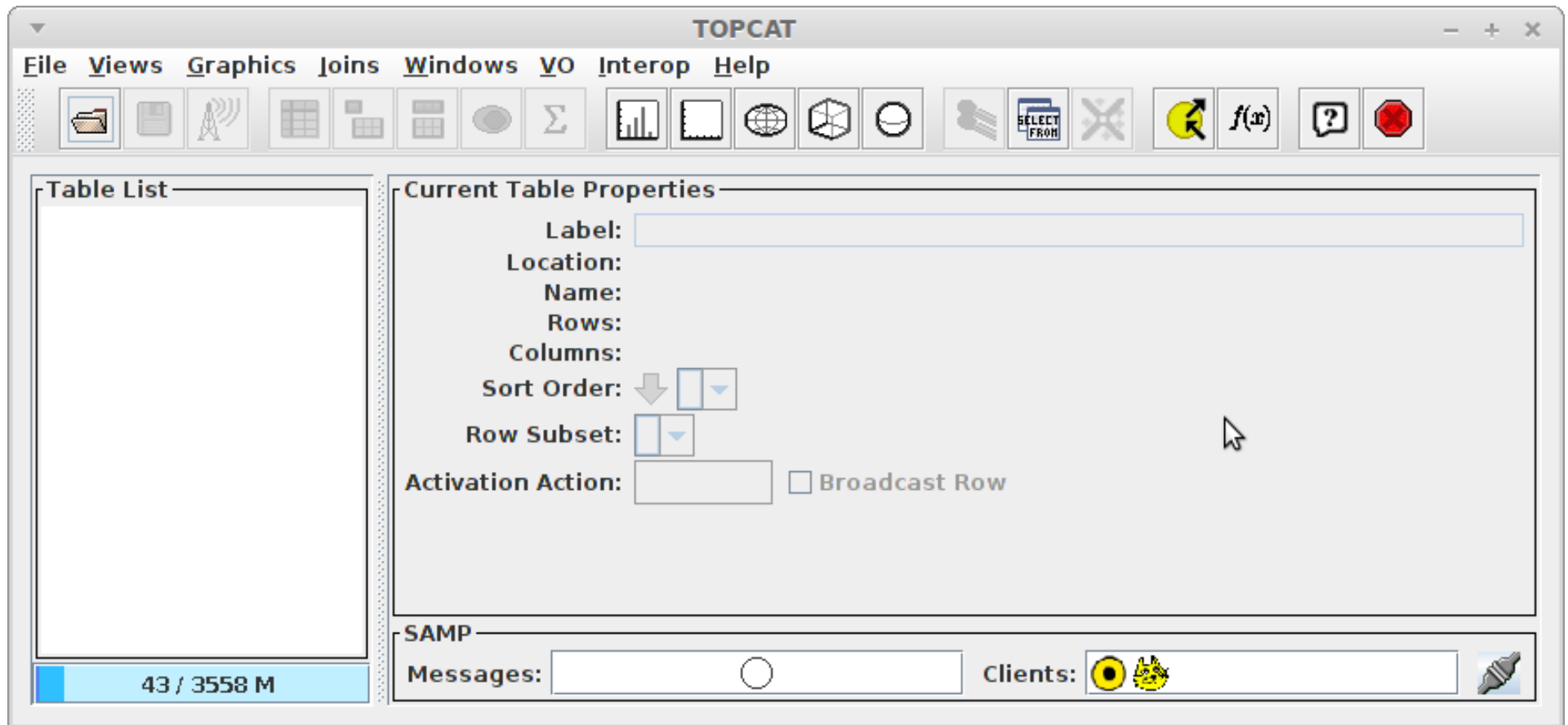
```
# EXERCISE
# Retrieving the resolved objects with a local surface density
# within 1' of resolved objects larger than 10000 objects/deg^2
SELECT TOP 100 t1.number as id,
    t1.ALPHA_J2000 as RA, t1.DELTA_J2000 as DEC,
    COUNT(*) as n_objects,
    COUNT(*)/AREA(CIRCLE('ICRS',t2.ALPHA_J2000,t2.DELTA_J2000,1/60.))
    as surf_density_neighbours
FROM
    jplus.MagABDualObj t1,
    jplus.MagABDualObj t2
WHERE
    t1.tile_id=2035 AND t2.tile_id=2035 AND
    t1.CLASS_STAR<0.8 AND t2.CLASS_STAR<0.8 AND
    1=CONTAINS(POINT('ICRS',t2.ALPHA_J2000,t2.DELTA_J2000),
        CIRCLE('ICRS',t1.ALPHA_J2000,t1.DELTA_J2000,1/60.))
GROUP BY
    t1.number,t1.ALPHA_J2000,t1.DELTA_J2000
HAVING
    surf_density_neighbours>10000.
ORDER BY
    surf_density_neighbours
DESC
```

Not Web-based Access to Archive

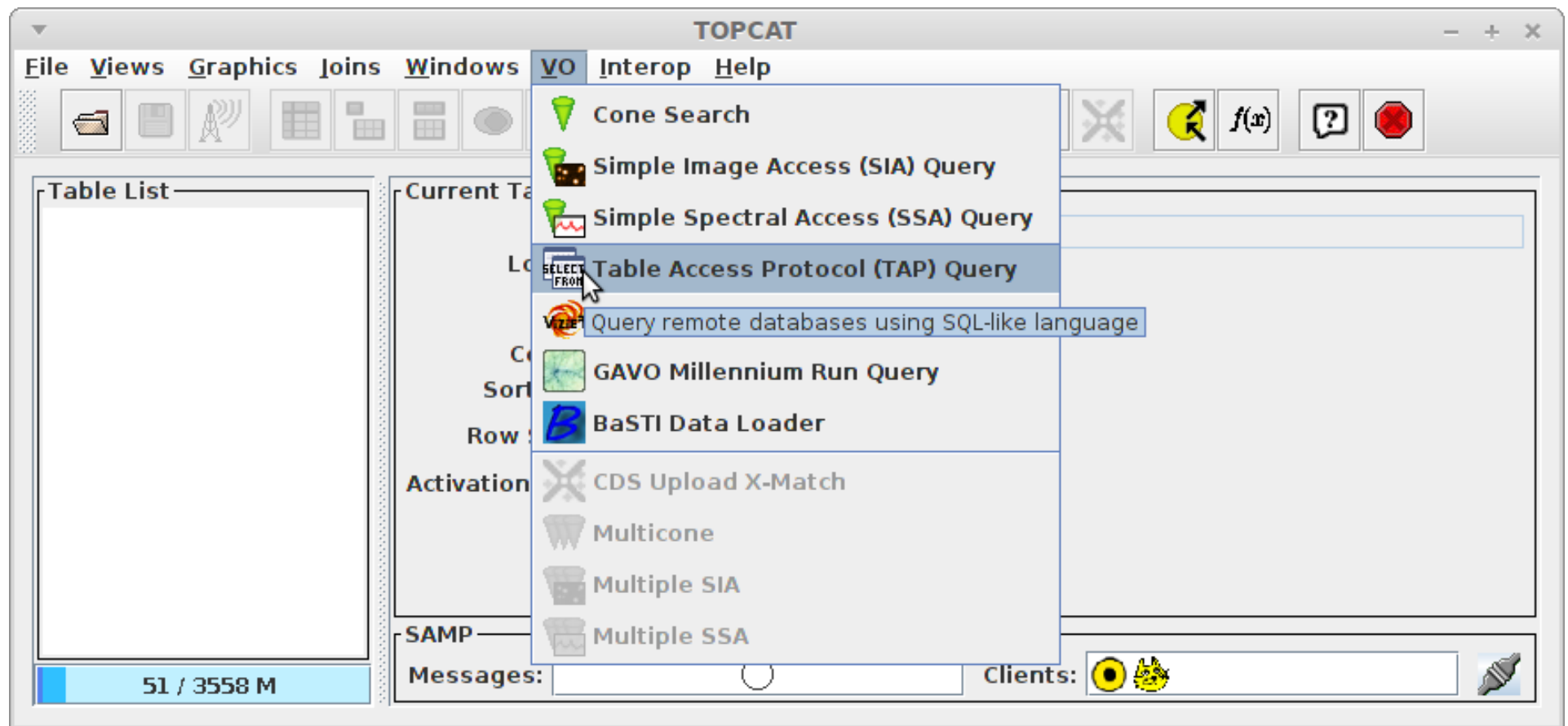
Not Web-based Access to Archive

- The J-PLUS Archive offered a series of V.O. Services:
 - Cone-search (SCS)
 - Tables (TAP)
 - Images (SIA)
 - Photo-spectra (SSAP)
- Any application including TAP service can access to the J-PLUS Archive and send ADQL Queries. Eg.:
 - TOPCAT
 - STILTS
 - wget/curl
 - Python
- It is also possible to access the FITS images throught the SIA Protocol using Aladin.

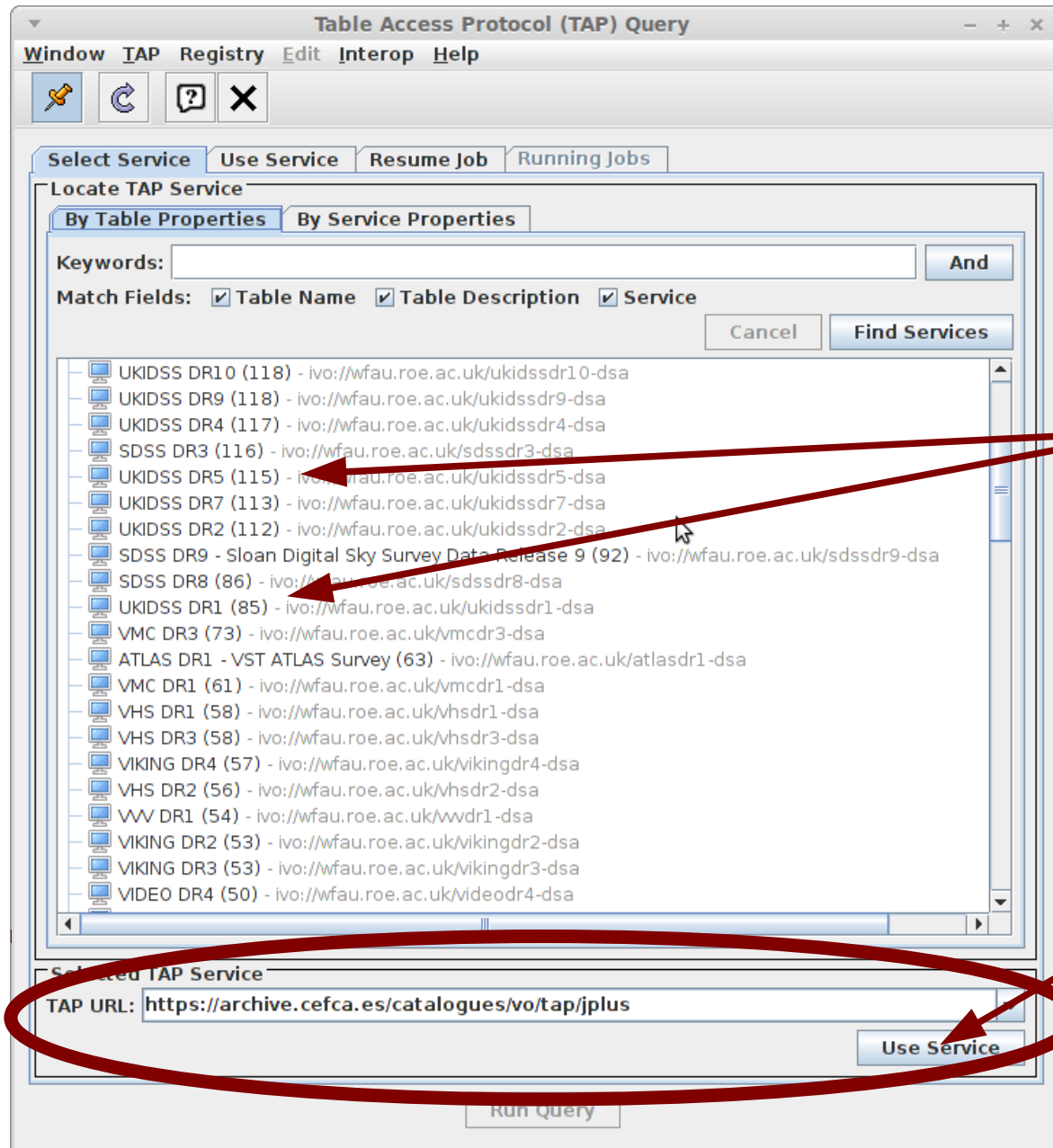
TOPCAT



TOPCAT



TOPCAT



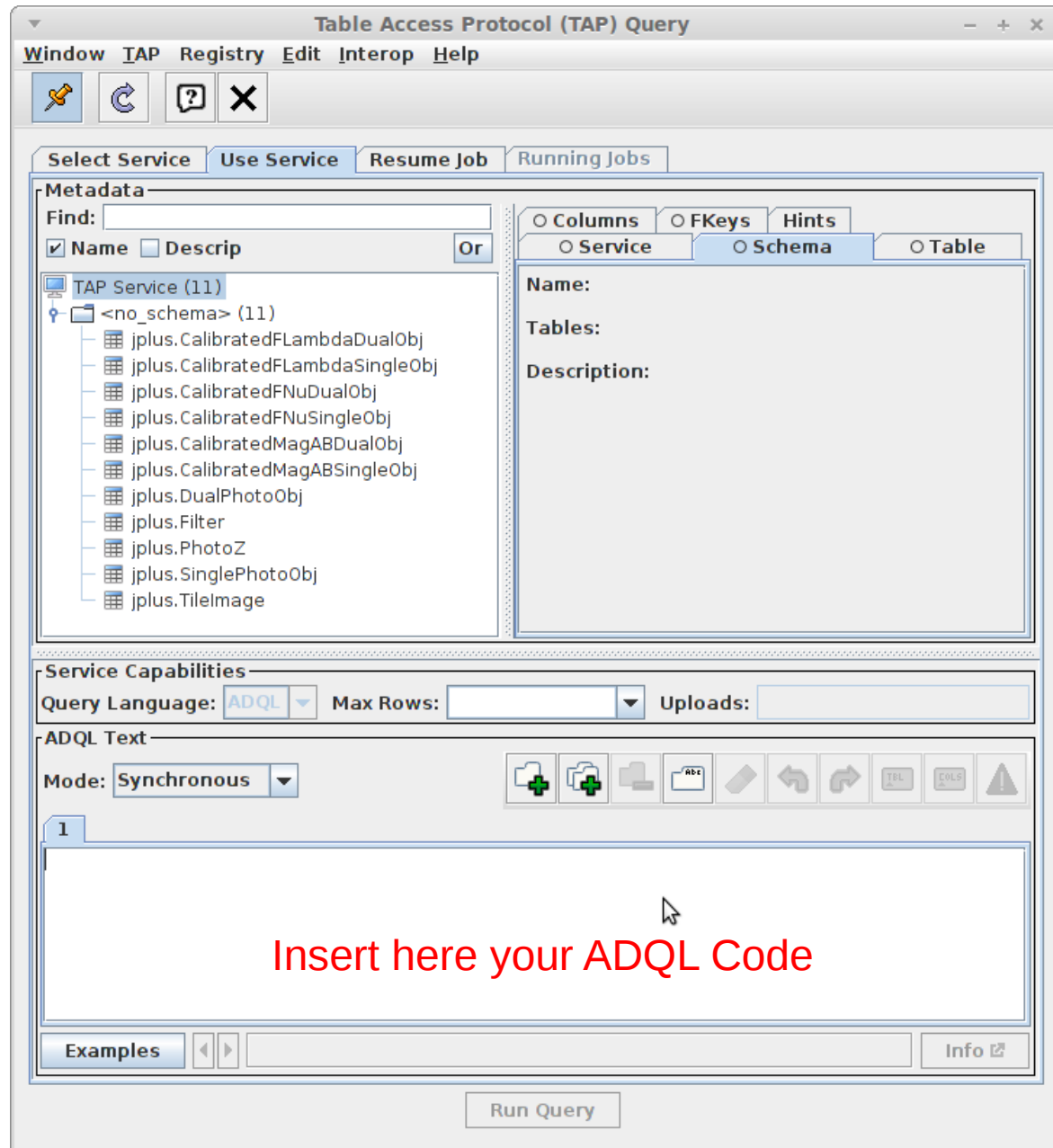
Select to use any
of the listed
archives

To use J-PLUS
Archive

JAVA Certificate Issue

- See e-mail by David Cristóbal, titled “Some maps with J-PLUS, Sloan offset in photometry”, on Sep 27th, 2016.
- Retrieve CEFCA certificate:
 - `curl http://www.cefca.es/CA_CEFCA_Root.crt > CA_CEFCA_Root.crt`
- Locate JAVA folder:
 - `locate security/cacerts`
- Add certificate (change JAVA's path to your system's):
 - `sudo /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/keytool -import -trustcacerts -alias CAFcefca -file CA_CEFCA_Root.crt -keystore /usr/lib/jvm/java-7-openjdk-amd64/jre/lib/security/cacerts`

TOPCAT



STILTS

```
$ more do_query_stilts.sh
#!/bin/bash
url=http://upad.cefca.es/catalogues/vo/tap/jplus
user=$1
shift
query=$@
unset pass
function getpass {
    if [ -z "${pass+x}" ];
    then
        read -s -p Password: pass
    fi
}
getpass
stilts -Dstar.basicauth.user=${user} \
      -Dstar.basicauth.password=${pass} tapquery\
      tapurl=${url}\
      adql="${query}" ofmt=csv

$
$ ./do_query_stilts.sh 'SELECT name FROM jplus.TileName'
```

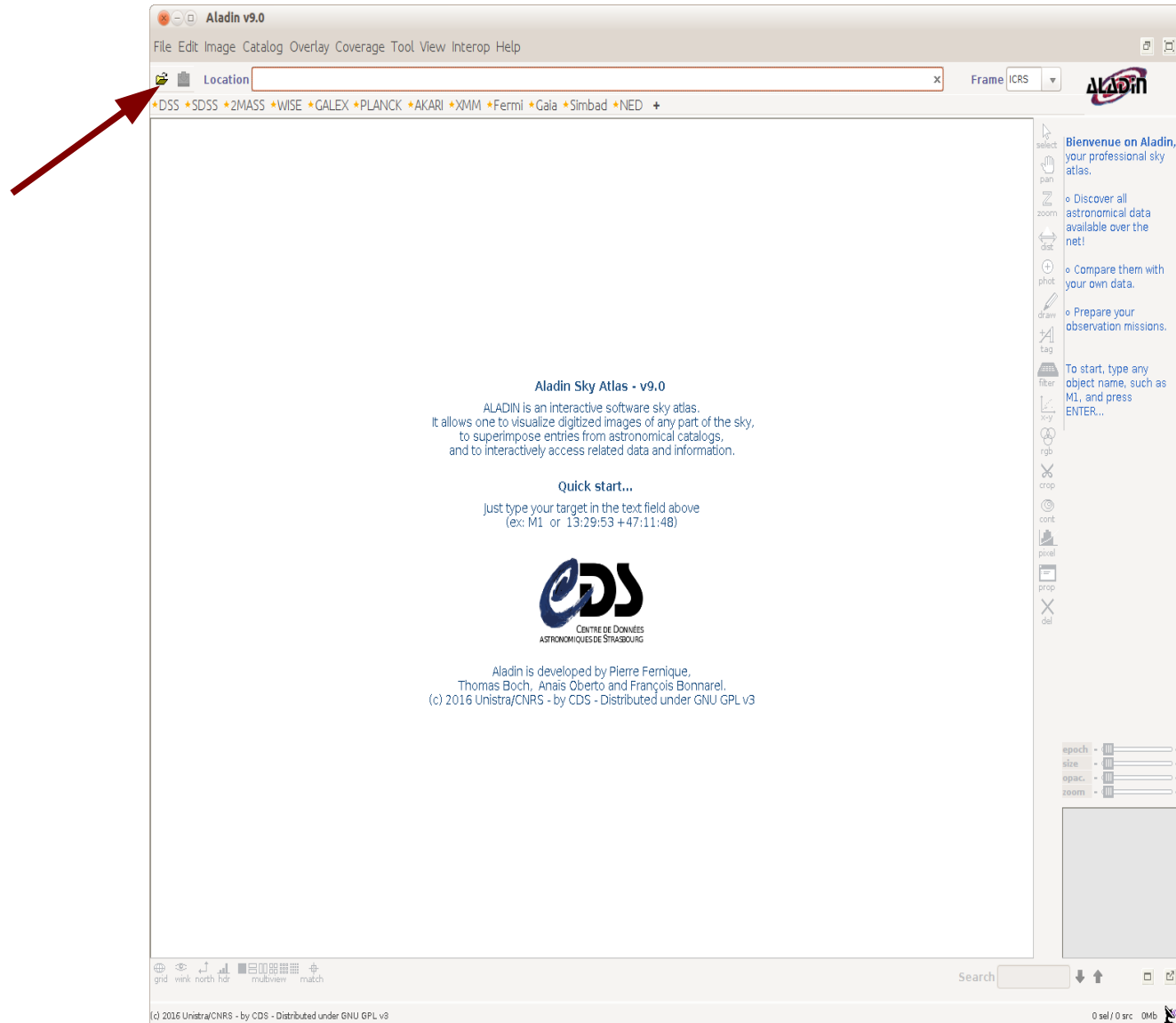
CURL

```
$ more ./do_query.sh
#!/usr/bin/env bash

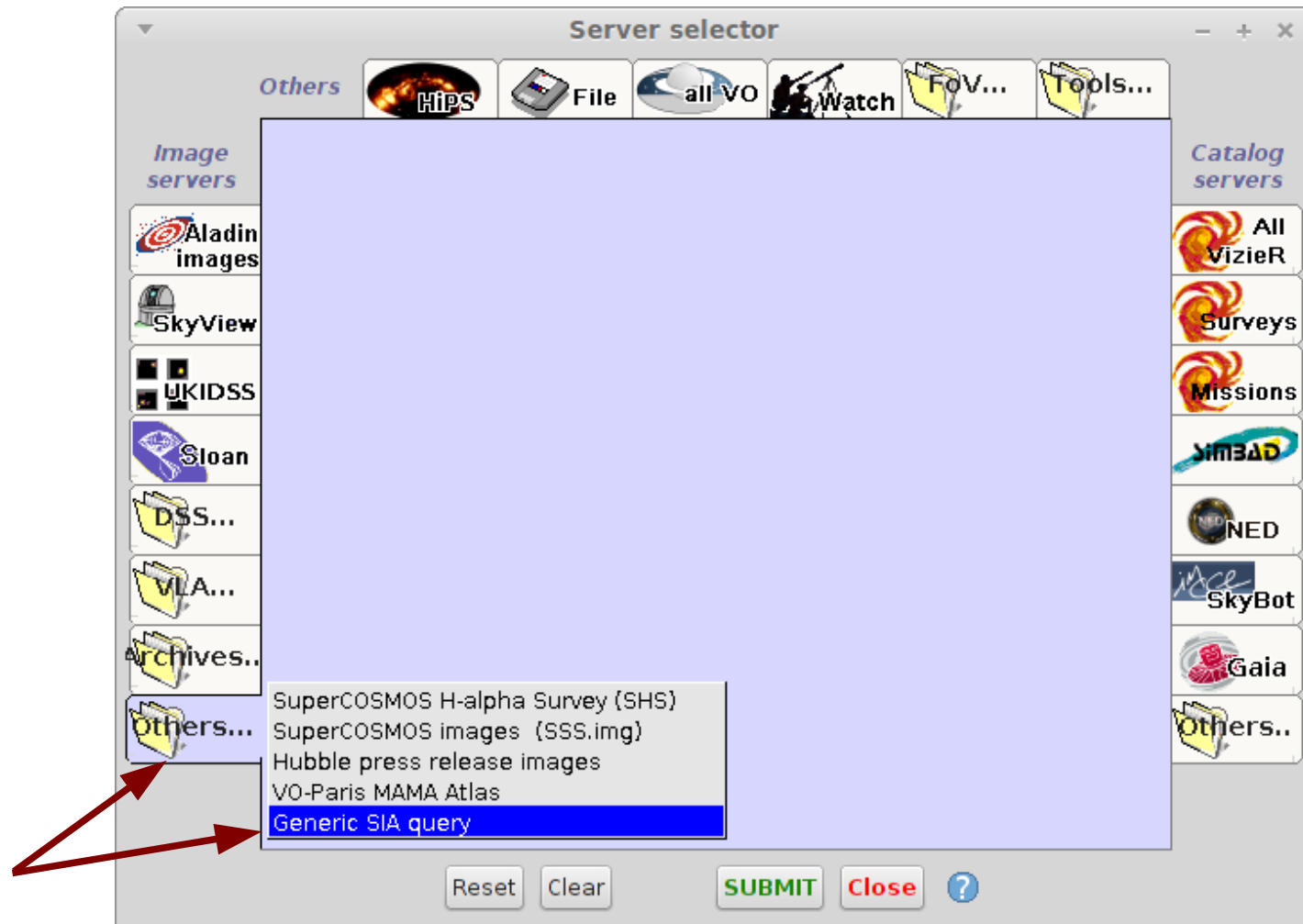
url="https://archive.cefca.es/catalogues/vo/tap/jplus"
user=$1
shift
query=$@
curl -u ${user} --request POST \
    --location \
    --data REQUEST=doQuery \
    --data PHASE=RUN \
    --data FORMAT=votable \
    --data LANG=ADQL \
    --data "QUERY=$query" \
    --insecure \
    $url

$
$ ./do_query.sh 'SELECT name FROM jplus.TileName'
```

Aladin



Aladin



Aladin

Insert target and search radius

The screenshot shows the 'Server selector' window in Aladin. It features a top toolbar with icons for 'Others', 'HiPS', 'File', 'all VO', 'Watch', 'FoV...', and 'Tools...'. The main area is titled 'Generic SIA query' and contains three input fields: 'Target (ICRS, name)' with the value '120.3 40.2', 'Radius' with the value '14'', and 'Base URL (incl. '?')' with the value 'https://archive.cefca.es/catalogues/vo/siap/jplus?'. A 'Grab co...' button is next to the target field. Below these fields is a list of search results, including 'gSDSS', 'iSDSS', 'J0395', 'J0410', 'J0430', 'J0515', and 'J0660', each with a checkbox and a description of the survey and field size. The left sidebar shows 'Image servers' with icons for 'Aladin images', 'SkyView', 'UKIDSS', 'Sloan', 'DSS...', 'VLA...', 'Archives...', and 'Others...'. The right sidebar shows 'Catalog servers' with icons for 'All VizieR', 'Surveys', 'Missions', 'SIMBAD', 'NED', 'iACe', 'SkyBot', 'Gaia', and 'Others..'. At the bottom are buttons for 'Reset', 'Clear', 'SUBMIT', 'Close', and a help icon. Red arrows point from the text annotations to the 'Target' and 'Radius' fields, the 'Base URL' field, and the 'SUBMIT' button.

Insert SIAP URL ending in "?"

Press to get list and then to load selected images