International
Virtual
Observatory
Alliance

IVOA SCS: Simple Cone Searches for Astronomical Catalogues

Version 1.0

IVOA Working Draft 2025-12-01

Working Group
DAL
This version
https://www.ivoa.net/documents/SCS2/20251201
Latest version
https://www.ivoa.net/documents/SCS2
Previous versions
This is the first public release
Author(s)
Markus Demleitner
Editor(s)
Markus Demleitner
Version Control
Revision d7e138f, 2025-10-01 10:57:12 +0200

Abstract

The Simple Cone Search Protocol is an IVOA protocol designed to support publishing astronomical catalogues to the Virtual Observatory with modest implementation requirements, and querying them using any language that has an http library and a VOTable parser.

This document defines query parameters, response formats, and metadata standards. The present specification is incompatible with the IVOA's ConeSearch version 1 protocol. We therefore also propose a plan for how to transition between the two major versions.

Status of this document

This is an IVOA Working Draft for review by IVOA members and other interested parties. It is a draft document and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use IVOA Working Drafts as reference materials or to cite them as other than "work in progress".

A list of current IVOA Recommendations and other technical documents can be found in the IVOA document repository[1].

Contents

[1] https://www.ivoa.net/documents/

Conformance-related definitions

The words "MUST", "SHALL", "SHOULD", "MAY", "RECOMMENDED", and "OPTIONAL" (in upper or lower case) used in this document are to be interpreted as described in IETF standard RFC2119 (Bradner, 1997).

The *Virtual Observatory (VO)* is a general term for a collection of federated resources that can be used to conduct astronomical research, education, and outreach. The International Virtual Observatory Alliance (IVOA) is a global collaboration of separately funded projects to develop standards and infrastructure that enable VO applications.

1 Introduction

In 2008, Simple Cone Search (SCS) version 1 (Plante and Williams et al., 2008), hereafter referred to as ConeSearch-1[2], was one of the first standards in the Virtual Observatory to gain significant traction; there was a standardised way to query astronomical catalogues – tabular data with celestial positions and a simple primary key – across data centres.

In addition to taking up previous standards efforts, it was exploring various technologies, such as a minimal data model annotation using UCDs, bespoke error reporting patterns, and metadata extensions for the VO Registry; at least within the VO, this was all breaking new ground.

Not surprisingly, in the 18 years since ConeSearch-1, many of these approaches turned out to be insufficient; in particular the completely outdated form of ConeSearch-1's UCDs make it stick out like a sore thumb in today's Virtual Observatory. But in many other respects, from error messages to metadata discovery, SCS should respect the general rules for protocols defined by the Data Access Layer Working Group, DALI (Dowler and Demleitner et al., 2017).

Since ConeSearch-1's release, the Table Access Protocol TAP (Dowler and Rixon et al., 2019) has been defined. Thus, there is now a far superior protocol to query tabular data. However, a niche for a cone search-like protocol still exists: something that, on the server side, can be implemented ad-hoc, and on the client side is dead simple to use provided you have a VOTable parser.

The remainder of this document first explores the use cases in a bit more detail, then defines the endpoints required for an SCS2 service and goes on to specify the response format. The main part concludes with considerations of how to register, discover, and query SCS2 services. This is complemented by an appendix giving recommendations on how the standard authors expect clients to discover and query resources with SCS2 interfaces (and what they

[2]The move in the naming scheme from ConeSearch-1 to SCS2 has no deeper meaning; it just follows the scheme established by later "S-Protocols" (SIAP, SSAP, SLAP...)

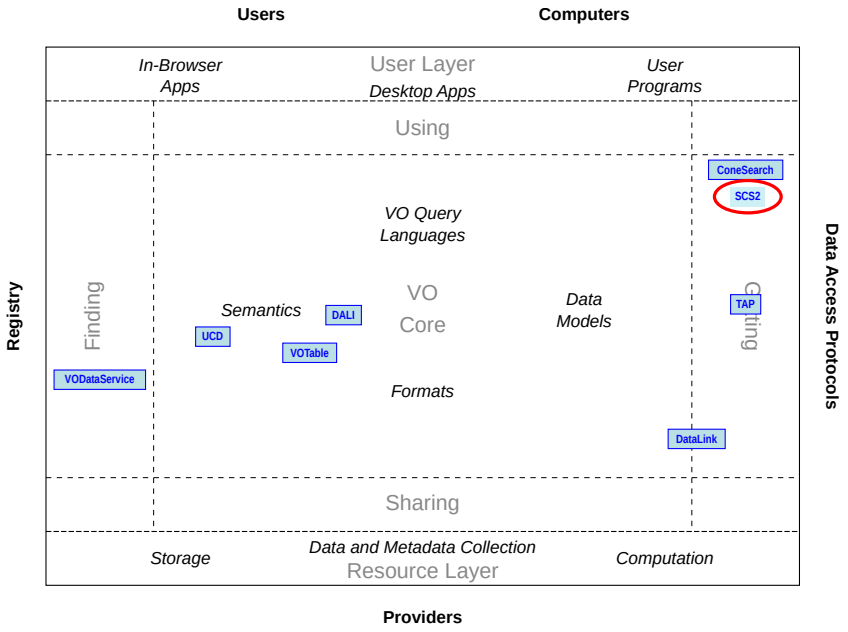

Figure 1: Architecture diagram for this document

would rather not see). Finally, we sketch a migration plan to completely replace the ConeSearch-1 ecosystem in a manner that should reduce breakage in running services as much as possible.

1.1 Role within the VO Architecture

Fig. 1 shows the role this document plays within the IVOA architecture (Dowler and Evans et al., 2021).

SCS2 directly makes use of the following standards:

DALI
: Dowler and Demleitner et al. (2017) gives the basic rules all modern DAL protocols have to follow. We will freely reference DALI in many places. The expectation is that any minor version of DALI 1 should yield compatible services, although chapter-sharp references in this document will only be accurate for the exact version DALI 1.2.

VOTable
: Ochsenbein and Taylor et al. (2025) defines the central response format for SCS2 services. Actually, most other possible response formats are missing metadata required for proper SCS operations; in that sense, there is no SCS2 without VOTable.

UCD
: Derriere and Preite Martinez et al. (2005) gives a lightweight semantic annotation at allows clients to (semi-) automatically process SCS2 results.

VODataService
: Demleitner and Plante et al. (2021) is the metadata scheme used to register SCS2 services.

SCS2 also relates to the following other VO standards:

ConeSearch 1
: Plante and Williams et al. (2008) is the incompatible predecessor of SCS2. We give a plan for how to transition from it in this document.

TAP
: Dowler and Rixon et al. (2019) is the IVOA's protocol for running advanced queries against tables of all sorts. Its existence allows us to ignore more advanced use cases by defering them to TAP.

DataLink
: Bonnarel and Dowler et al. (2023) offers a mechanism to link advanced data products, in particular parts of a catalogue row's provenance chain, to rows returned by SCS2.

2 Use Cases

2.1 Consumer Use Case

SCS2 is driven by the main use case: Researchers wants to retrieve a full astronomical catalogue or parts of it and immediately start working with it, without having to write importing software, and ideally with support in quickly adapting to different catalogues. The basic specification of what part of a catalogue is by position; being able to formulate further simple constraints is a nice extra, but SCS2 does not attempt to compete with TAP. Still, clients should be able to produce expressive user interfaces for SCS2 services on the fly.

A related second use case is: Researchers want to locate a catalogue with certain properties (e.g., the presence of a column covering a certain sort of physics, some limiting magnitude, spatial or spectral coverage). On finding a matching resource, they can immediately retrieve data pertaining to their research (and preferably not terribly much more).

To briefly map these use cases to features covered by this standard:

- Immediately start working: VOTable response format
- Positional contraint: The mandatory RA, DEC, SR parameters

- Other parts of catalogues: The possibility to declare further parameters and their standard DALI interval syntax
- Locate a catalogue: Our use of the VO Registry, including registering tablesets containing UCDs.

2.2 Operator Use Case

A data publisher wants to make one or more catalogues accessible to the scientific public with minimal effort while still satisfying the consumer use case.

This important part in this use case is "with minimal effort". Without this requirement, TAP would be the protocol to choose, and data providers are urged to see whether they can adapt one of the existing TAP/ADQL implementations. However, implementing TAP and ADQL is a major effort. SCS2's goal is to be implementable within a week or so even without a relational database.

3 Endpoints

SCS2 is what DALI (Dowler and Demleitner et al., 2017) calls a "concrete service specification". As envisioned in DALI, we will only reference DALI sections where they apply unchanged. The references are for DALI 1.2.

All SCS2 services must implement three endpoints with the names given here:

- `<base-url>/scs2` – synchronous query. This is the query endpoint further specified below. Its full URL is what is registered as "service URL". All parameters on this endpoint are case-sensitive.
- `<base-url>/capabilities` – VOSI capabilities as per DALI sect. 2.5. This must describe the SCS2 and VOSI capabilities but may contain arbitrary other capabilities.
- `<base-url>/tables` – VOSI tables as per DALI sect. 2.6. The names of all tables in the tableset returned must work as values for the TABLE parameter of the SCS2 endpoint.

Note that this scheme in particular means that a client can reliably infer the URL of the VOSI capabilities by computing the sibling of the service URL named "capabilities".

4 Parameters in SCS2

SCS2 services are required to fail when requests pass parameters they do not support, regardless of whether values are bound to these parameters or not.

4.1 Standard Parameters

As in ConeSearch-1, SCS2 services must support positional constraints. They are given as a "cone", i.e., a centre and a search radius. The following parameters – backwards-compatible with ConeSearch-1 – are required for that:

RA – a right ascension in degrees, understood to be ICRS at whatever epoch the catalogue is in.

DEC – a declination in degrees, understood to be ICRS at whatever epoch the catalogue is in.

SR – a search radius in degrees.

All these parameters are not repeatable.

When evaluating the cone constraint, services must compute spherical distances to the cone center and only include objects closer to the centre than SR. For the benefit of non-SCS2 clients, services should declare these input parameters with the UCDs *pos.eq.ra*, *pos.eq.dec*, and *pos.angDistance*, respectively.

SCS2 breaks the identity of service and catalogue that ConeSearch-1 had. Services therefore must support the input parameter `TABLE`. This is not repeatable, either.

The value of `TABLE` is a table name taken from the tableset. Services only offering access to a single table may accept requests without a `TABLE` parameter but must accept it and raise an error if the table name does not match what they give in their tableset. For services publishing multiple tables, it is an error to not pass `TABLE`.

All SCS2 services must support the `RESPONSEFORMAT` parameter as per DALI sect. 4.3.3. The only required response format is VOTable, where implementations are free to return any version with a major version number of 1. VOTables must be selectable as with all media types given in DALI. Without `RESPONSEFORMAT`, SCS2 services must return VOTable.[3]

All SCS2 services must support the `MAXREC` parameter as per DALI sect. 4.3.4.

All SCS2 services must support a `VERB` (for "verbosity") parameter, which we keep from ConeSearch-1. Its value must be one of 1, 2, or 3. When the

[3] In other words, a totally sensible and legal implementation of `RESPONSEFORMAT` is to raise an error if anything but application/x-votable+xml or text/xml is passed to it and ignore the parameter otherwise.

value is 1, the response should include the bare minimum of columns that convey some basic information kept in the catalogue. When the value is 3, the service should return all columns making up the underlying table. A value of 2 requests the columns considered by the provider to be most typically useful to the user. It is legal to always return the same set of columns independent of the value of `VERB`.

SCS2 service may support a `POS` parameter as defined by SIAP2 (Dowler and Bonnarel et al., 2015), except that for SCS2, this is a non-repeatable parameter.

SCS2 services may support an `UPLOAD` parameter as per DALI sect. 4.3.5. Only a single table in VOTable format may be uploaded, and its columsn must be named RA, DEC, and SR. Services implementing `UPLOAD` must produce a VOTable with a single result table containing the union of a series of SCS queries with the cones defined by the rows of the uploaded table. Duplicate rows are not allowed. Table uploads from http(s) URIs and inline uploads must be supported.

4.2 Free Parameters

Services may offer additional query parameters to allow clients to communicate further constraints. To constrain float-like parameters, use interval-typed input parameters as per DALI sect. 3.4.

Services should name the parameters like the table columns they constrain. If at all possible, names of free parameters should be in all-lower case. This is to make them reliably distinct from protocol parameters, which are always upper case; it is still stronly discouraged to have free parameters that after case folding clash with protocol parameters.

4.3 Error Conditions with Parameters

When clients pass parameters to an SCS2 services that it does not support, it must respond with a 400 Bad Request HTTP status code and a DALI error message explaining which parameter caused the problem.

When clients pass multiple values for single-valued parameters, the service must respond with a 400 Bad Request HTTP status and a DALI error message explaining what parameter was duplicated.

When clients pass potentially conflicting parameters to an SCS2 service, such as both RA, DEC, and SR one the one side and POS on the other, or UPLOAD and any other positional specification, they must respond with a 400 Bad Request HTTP status and a DALI error message explaining what conflict caused the error.

5 Protocol Responses

SCS2 services respond to requests as defined in DALI sect. 5. The following additional requirements apply:

- Error responses should not use a 200 HTTP status code but instead 4xx when the service sees a client error and a 5xx when the service diagnoses an error on its side.
- The media types text/xml and application/x-votable+xml are treated exactly equivalent by SCS2 clients and indicate a VOTable response.
- All responses, including errors, must be in VOTable unless the client has requested a different `RESPONSEFORMAT`, in which case none of the following constraints apply.
- Exactly one *`FIELD`*, the row identifier, must have a UCD of *meta.id;meta.main* and must be an array of VOTable chars; this applies even if the original catalogue uses integer identifiers. This must be suitable as a primary key, i.e., there must not be two rows sharing the same row identifier.
- Exactly one *`FIELD`* must have a UCD of *pos.eq.ra;meta.main* and must be a single floating point value (float or double); its contents represents a right ascension in the ICRS frame.
- Exactly one *`FIELD`* must have a UCD of *pos.eq.dec;meta.main* and must be a single floating point value (float or double); its contents represents a declination in the ICRS frame.
- All *`FIELD`* elements must have a description, and for quantities with units, the units must be given in VOUnits syntax (Derriere and Gray et al., 2014).

The epoch of the main position should be defined in a *`COOSYS`* element.

6 Registration of SCS2 Services

SCS2 services are registered as VODataService (Demleitner and Plante et al., 2021) *`vs:CatalogService`* records. They must come with a tableset; empty or missing *`colum/description`* elements are not allowed in this tableset. SCS2 services also must declare a spatial coverage and should declare spectral and temporal coverage as appropriate.

Registry records of SCS2 services must have at least one capability with the standard id

ivo://ivoa.net/SCS2#query-2.0

Such a capability must contain exactly one interface of type *vs:ParamHTTP* with its *role* set to *std*. All accepted parameters, including the mandatory ones, must be declared within this interface element. Even for SCS2-defined parameters, empty or missing param/description elements are not allowed. All free parameters should come with a UCD.

The capability element with the SCS-2.0 *standardID* should be of the type *cs:ConeSearch* defined below.

Other capabilities are allowed; in particular, the VOSI capabilities (which themselves are mandatory) should be declared in the registry record as shown in DALI, sect. 2.5. Where the SCS2-published table is also published through a TAP services, an TAP auxiliary capability should be given as specified in Demleitner and Taylor (2019) (DDC).

There are two major publication scenarios:

1. Where an SCS2 service publishes only a single table, publishers should use a single *vs:CatalogService* record containing the tableset and the full capability.

2. Where a data centre has multiple tables available for SCS2, and a generic service without extra free parameters handles them, the preferred scenario is to have a single *vs:CatalogService* record declaring the capability, including the common parameters, as well as the tableset. In addition, for each table, there is a *vs:CatalogResource*-typed record with an auxiliary capability and a relationship referencing the service record.

Note that services supporting additional free parameters will usually have to use pattern (1).

In the second case, the auxiliary SCS2 capability will not use the *cs:ConeSearch* type; at this time, we recommend a plain *vr:Capability*-typed capability. Its standardID must be

ivo://ivoa.net/SCS2#query-2.0-aux

Example registry records for both scenarios are available in Appendix C.

A Usage Patterns (non-normative)

In general, it is discouraged to for clients to do registry queries constraining to SCS2 services when what their users presumably want is data discovery; see the DDC note Demleitner and Taylor (2019) for a discussion of service vs. data discovery.

Hence, the typical usage pattern of a science client[4] would look like this:

[4] "science" as opposed to "infrastructure"; a validator might have every reason to look exclusively for SCS2 capabilities, for instance.

- Perform a registry search with non-capability constraints (e.g., on keywords, UCDs, authors). In the query, retrieve the available capabilities, e.g., using `ivo_string_agg` UDF.
- If there is a capability with one of SCS2's standard ids (either query or query-aux), offer a UI to use the service or immediately run a simple cone search based on the the mandatory parameters (RA, DEC, SR).
- To produce a UI, fetch the `capabilities` sibling of the access URL discovered. Enumerate the capability elements with an SCS2 standard identifier.
  - If there is exactly one non-aux capability, expose the declared parameters to the user[5]
  - If there is only one aux capability, get the access URL from that capability's interface element, retrieve its capabilities document and build the API or UI as described here.

Against that, clients explicitly planning to have an "SCS2 client" (and that is mildly discouraged) would probably want to do the following:

- Perform a registry search with all normal constraints, additionally constraining `standard_id` in `rr.capability` via a constraint like

  ```
  standard_id like 'ivo://ivoa.net/std/SCS2#query-2.%'
  ```

  Also retrieve the relationships of the matches. The wildcard is necessary to match records having auxiliary capabilities; these contain relevant metadata like descriptions, authors, coverage, etc. Also retrieve the relevant table names.
- To avoid querying the same tables multiple times, remove any matches with an ivoid that is also mentioned in the related resources (these will be the main SCS services).
- Either run simple cone queries against the services and tables found in this way, passing to TABLE the table name(s) discovered, or let users select individual services; in that latter case, try to build APIs with any extra parameters as above, interpreting the service's capabilities endpoint.

[5] At the time of writing, we are still missing an interoperable method to declare column statistics, which admittedly are important to build meaningful UIs. Demleitner and Mantelet (2021) suggests a conceivable mechanism, and the authors expect that a similar scheme will be adopted soon in VODataService, at which point these statistics should be exposed to users or APIs.

Here is an example for a RegTAP query that will yield SCS2 services serving temperatures that would work for this kind of SCS2 UI:

```
SELECT ivoid, access_url, res_title, res_description,
  related_ids
FROM rr.capability
  NATURAL JOIN rr.interface
  NATURAL JOIN rr.resource as b
  NATURAL LEFT OUTER JOIN (
    SELECT ivo_string_agg(related_id, '###sep###') AS related_ids
    FROM rr.relationship
    WHERE relationship_type='isservedby'
    GROUP BY ivoid) AS rels
WHERE
  standard_id LIKE 'ivo://ivoa.net/std/scs2#query-2.%'
  AND EXISTS(SELECT 1 FROM rr.table_column AS t
    WHERE t.ivoid=b.ivoid AND ucd='phys.temperature')
```

For clarity: Clients only wishing to do a plain cone search on a single service or some hand-curated subset of SCS services do not have to retrieve the capabilities document or do complicated elision of collective services or do other sorts of complex service discovery. A functionality of the type "query this list of SCS2 services for this particular cone" is of course completely in the spirit of the VO. Still, SCS2 clients are encouraged to think more about data discovery than about service disovery.

Also note that there is no defined way to produce a union of SCS2 results produced querying multiple SCS2 services. Trivially, their table schemas will be different. But one cannot even safely produce a union of the guaranteed columns (main identifier, RA, and Dec); for instance, the uniqueness requirements of the identifiers would be lost in this way.

B Transition Plan (non-normative)

It is the express intent of this standard to completely replace ConeSearch-1 in the Virtual Observatory. Actually, prototyping the management of a major version transition was part of the motivation for producing this document. Whether transition plans for future major version transitions should be part of the respective standards or should come as external documents (e.g., IVOA notes) remains to be seen. For this prototype, however, the editor suggests that this plan should be part of the citable, permanent record that IVOA RECs comprise.

Desiderata of the managed transition[6] include

[6] adapted from https://github.com/ivoa/major-version-transition.

- Users should not see different VOs depending on which client they use
- Services should not be required to implement multiple major versions for longer than absolutely necessary to ensure a halfway smooth transition
- There cannot be flag days when all services must switch from version A to version B
- Clients should not be required to impement multiple major versions
- A client written in year Y should not break because of the transition before year Y+N, where N is perhaps 3 or so.

In order to attain these goals as well as possible in a global, federated system run with often rather minimal budgets, we propose the following time line; units are years, the time symbols are explained below.

$T_{\text{final-WD}}$ When the DAL WG declares that the standard is ready for adoption, several major data centres start implementing and registering SCS2 services; in the run-up to the final WD, pledges have been collected from: GAVO Heidelberg Data Centre.

Further Pledges?

T_{RFC} RFC is started when there are SCS2 services from at least five different publishers. The editors start actively soliciting client implementations.

T_{REC} The TCG will not promote SCS2 to REC before (at least basic) client implementations are available in pyVO, TOPCAT, and Aladin. At this point, an Erratum process is started to add a note to ConeSearch-1 stating that it is deprecated, but should still be offered even for new services for the next two years. Clients should prefer SCS2 if available.

A transition team is formed, consisting of members of at least three IVOA member projects; an ideal size would be five persons. At least one member each must be sent by one of the major searchable registries and one of the publishers with more than 100 ConeSearch-1 services.

At the beginning of the transition time, all SCS2 services must be still accompanied by ConeSearch-1.0 interfaces; this could be a problem for operators building new SCS server-side software during the transition time. When that situation arises, the transition team is free to apply common sense.

$T_{\text{REC}}+0.25$ The transition team is ready and takes over the management of the Erratum.

$T_{\text{REC}}+0.5$ The Erratum deprecating ConeSearch-1 is in place. The transition team starts monitoring SCS2 adoption: How many of the resources

with ConeSearch-1 capabilites have SCS2 capabilities, too? Which publishers do not offer SCS2 capabilities alongside yet? The transition team reports these numbers regularly to the TCG.

T_{REC}*+1* After one year, the transition team starts approaching publishers that operate ConeSearch-1 services without SCS2 capabilities and offers them options to either update their software or migrate either their publishing operation to another software.

T_{p90} When at least 90% of the publishers of ConeSearch-1 services and at least 99% of the services have SCS2 counterparts, this is noted on the interop list. New versions of cone search clients should from now on issue warnings when ConeSearch-1 interfaces are being used.

T_{p90}*+1* At least one major data centre starts monitoring the use of ConeSearch-1 services and the clients that issue requests against them. From this, an impact assessment is derived: how many users are on clients that do not know how to deal with SCS2? What clients are these? The transition team should investigate means to reach out to users or communities with particular migraton problems, for instance by organising "Migration weeks" when several major data centres serve errors to ConeSearch-1 clients.

The transition team continues engaging with publishers of ConeSearch-1 services to provide SCS2 endpoints alongside of them.

The transition team also prepares an IVOA note reporting on the current state and the transition process so far, in particular comparing the real run of events with this timeline.

T_{p90}*+3* Provided traffic on the ConeSearch-1 endpoints is negligible, an Erratum to ConeSearch-1 is prepared stating that the specification is now invalid and ConeSearch-1 is no longer a supported IVOA protocol. Data providers are encouraged to turn off their ConeSearch-1 endpoints, clients are encouraged to remove support for ConeSearch-1.

T_{p90}*+3.5* The searchable registries remove all ConeSearch-1 capabilities from their tables. The transition team updates its Note on the transition and then disbands.

Here is an explanation of the time variables used in the time line, and an optimistic estimate for what they might translate to (cf. Fig. 2):

$T_{\text{final-WD}}$ The time of the last working draft of the SCS2 standard; February 2026

T_{RFC} The time SCS2 RFC starts. May 2026

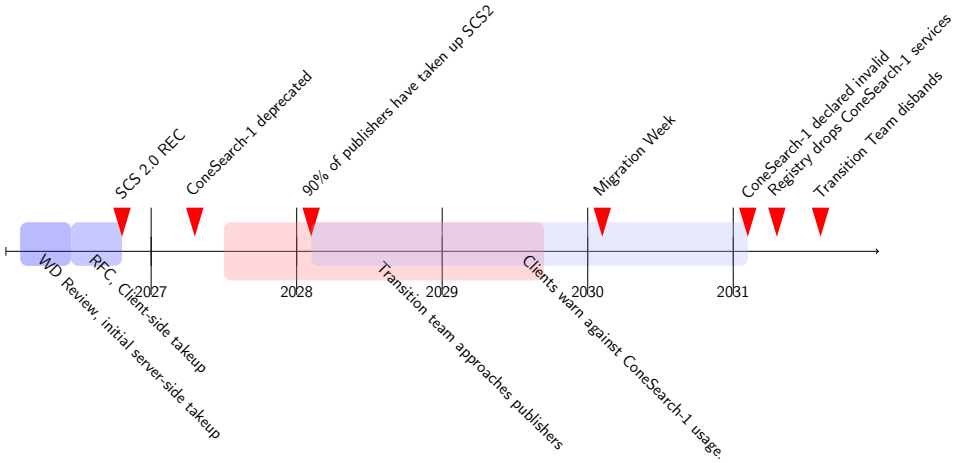

Figure 2: A graphical representation of the transition from ConeSearch-1 to SCS2 according to the optimistic scenario.

T_{REC} The time the TCG passes SCS2. November 2026

T_{p90} The time more than 90% of the publishers of ConeSearch-1 services have taken up SCS2. January 2028 (*if* the transition team actually functions)

In this optimistic scenario, we would turn off ConeSearch-1 in the summer of 2031.

Clearly, this idealised time line is not strictly binding; in the end, the transition team will have to react to how matters actually evolve. It is hoped that future transition plans can build upon the experiences made and reported by the SCS transition team.

C Example Documents

To help implementors, this document comes with several example documents:

- A sample response
- A registry record for a one-table service with custom parameters
- A registry record for a CatalogResource with an auxiliary SCS2 capability and the accompanying global SCS2 service declarations.

D Changes from Previous Versions

No previous versions yet.

References

Bonnarel, F., Dowler, P., Michel, L., Demleitner, M. and Taylor, M. (2023), 'IVOA DataLink Version 1.1', IVOA Recommendation 15 December 2023. doi:10.5479/ADS/bib/2023ivoa.spec.1215B, https://ui.adsabs.harvard.edu/abs/2023ivoa.spec.1215B.

Bradner, S. (1997), 'Key words for use in RFCs to indicate requirement levels', RFC 2119. http://www.ietf.org/rfc/rfc2119.txt.

Demleitner, M. and Mantelet, G. (2021), 'Towards blind discovery 2: Advanced column statistics', IVOA Note. http://ivoa.net/documents/Notes/colstatnote/.

Demleitner, M., Plante, R., Stébé, A., Benson, K., Dowler, P., Graham, M., Greene, G., Harrison, P., Lemson, G., Linde, T. and Rixon, G. (2021), 'VODataService: A VOResource Schema Extension for Describing Collections, Services Version 1.2', IVOA Recommendation 02 November 2021. doi:10.5479/ADS/bib/2021ivoa.spec.1102D, https://ui.adsabs.harvard.edu/abs/2021ivoa.spec.1102D.

Demleitner, M. and Taylor, M. (2019), 'Discovering Data Collections Within Services Version 1.1', IVOA Endorsed Note 20 May 2019. doi:10.5479/ADS/bib/2019ivoa.rept.0520D, https://ui.adsabs.harvard.edu/abs/2019ivoa.spec.0520D.

Derriere, S., Gray, N., Demleitner, M., Louys, M. and Ochsenbein, F. (2014), 'Units in the VO Version 1.0', IVOA Recommendation 23 May 2014, arXiv:1509.07267. doi:10.5479/ADS/bib/2014ivoa.spec.0523D, https://ui.adsabs.harvard.edu/abs/2014ivoa.spec.0523D.

Derriere, S., Preite Martinez, A., Williams, R., Gray, N., Mann, R., McDowell, J., Mc Glynn, T., Ochsenbein, F., Osuna, P. and Rixon, G. (2005), 'An IVOA Standard for Unified Content Descriptors Version 1.10', IVOA Recommendation 19 August 2005, arXiv:1110.0525. doi:10.5479/ADS/bib/2005ivoa.spec.0819D, https://ui.adsabs.harvard.edu/abs/2005ivoa.spec.0819D.

Dowler, P., Bonnarel, F. and Tody, D. (2015), 'IVOA Simple Image Access Version 2.0', IVOA Recommendation 23 December 2015. doi:10.5479/ADS/bib/2015ivoa.spec.1223D, https://ui.adsabs.harvard.edu/abs/2015ivoa.spec.1223D.

Dowler, P., Demleitner, M., Taylor, M. and Tody, D. (2017), 'Data Access Layer Interface Version 1.1', IVOA Recommendation 17 May 2017. doi:10.5479/ADS/bib/2017ivoa.spec.0517D, https://ui.adsabs.harvard.edu/abs/2017ivoa.spec.0517D.

Dowler, P., Evans, J., Arviset, C., Gaudet, S. and Technical Coordination Group (2021), 'IVOA Architecture Version 2.0', IVOA Endorsed Note 01 November 2021. doi:10.5479/ADS/bib/2021ivoa.spec.1101D, https://ui.adsabs.harvard.edu/abs/2021ivoa.spec.1101D.

Dowler, P., Rixon, G., Tody, D. and Demleitner, M. (2019), 'Table Access Protocol Version 1.1', IVOA Recommendation 27 September 2019. doi:10.5479/ADS/bib/2019ivoa.spec.0927D, https://ui.adsabs.harvard.edu/abs/2019ivoa.spec.0927D.

Ochsenbein, F., Taylor, M., Donaldson, T., Williams, R., Davenhall, C., Demleitner, M., Durand, D., Fernique, P., Giaretta, D., Hanisch, R., McGlynn, T., Szalay, A. and Wicenec, A. (2025), 'VOTable Format Definition Version 1.5', IVOA Recommendation 16 January 2025. https://ui.adsabs.harvard.edu/abs/2025ivoa.spec.0116O.

Plante, R., Williams, R., Hanisch, R. and Szalay, A. (2008), 'Simple Cone Search Version 1.03', IVOA Recommendation 22 February 2008, arXiv:1110.0498. doi:10.5479/ADS/bib/2008ivoa.specQ0222P, https://ui.adsabs.harvard.edu/abs/2008ivoa.specQ0222P.