International
Virtual
Observatory
Alliance

MOC: Multi-Order Coverage map

Version 2.0

IVOA Recommendation 2022-07-27

Working group
Applications
This version
http://www.ivoa.net/documents/moc/20220727
Latest version
http://www.ivoa.net/documents/moc
Previous versions
Version1.1
Version1.0

Author(s)
Pierre Fernique (CDS), Ada Nebot (CDS), Daniel Durand (CADC), Matthieu Baumann (CDS), Thomas Boch (CDS), Giuseppe Greco (EGO-Virgo), Tom Donaldson (STScI/NASA), Francois-Xavier Pineau (CDS), Mark Taylor (University of Bristol), Wil O'Mullane (Vera C. Rubin Observatory), Martin Reinecke (Max Planck), Sébastien Derrière (CDS)
Editor(s)
Pierre Fernique, Ada Nebot, Daniel Durand

Abstract

This document describes the Multi-Order Coverage map method (MOC) version 2.0 to specify arbitrary coverages for sky regions and/or time coverages and potentially other dimensions. The goal is to be able to provide a very fast comparison mechanism between coverages. The mechanism is based on a discretization of space and time dimensions. The system is based on the definition of a specific storage of the map coverage using predefined cells hierarchically grouped which makes it easy to produce and use for exploring astronomical collections. There are already a few applications and libraries which are taking advantage of this major evolution of the MOC standard.

Status of This Document

This document has been reviewed by IVOA Members and other interested parties, and has been endorsed by the IVOA Executive Committee as an IVOA Recommendation. It is a stable document and may be used as reference material or cited as a normative reference from another document. IVOA's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability inside the Astronomical Community.

A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/Documents/.

Contents

List of Figures

List of Tables

Acknowledgments

This work has been supported by the ESCAPE project (the European Science Cluster of Astronomy & Particle Physics ESFRI Research Infrastructures) that has received funding from the European Union's Horizon 2020 research and innovation Programme under the Grant Agreement n. 824064.

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.

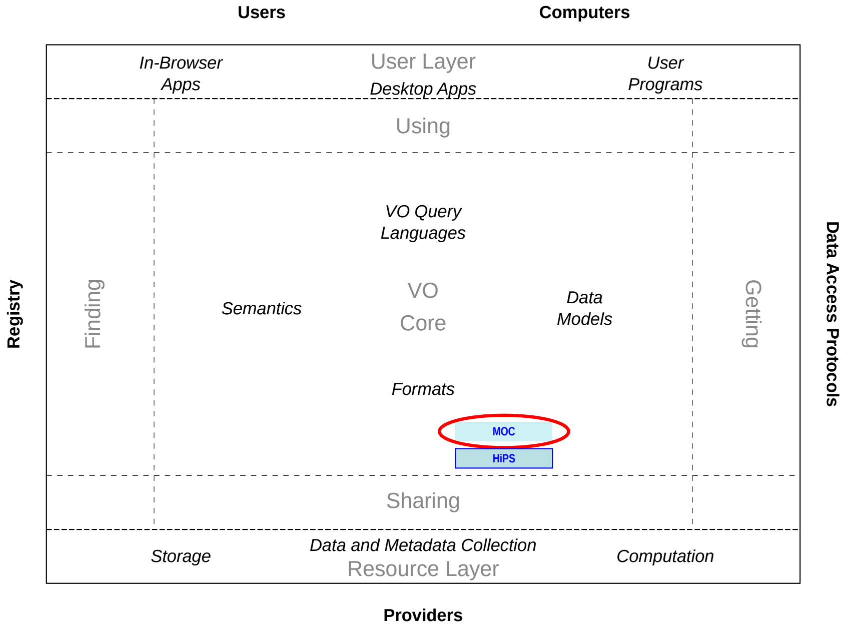

Figure 1: Architecture diagram for Multi-Order Coverage map.

1 Introduction

This document is a major release of the already existing encoding method recommendation *Multi-Order Coverage* map (MOC 1.1, Fernique and Boch et al., 2019). We generalize the MOC originally limited to space dimension (Space MOC, SMOC) to the time dimension (Time MOC, TMOC), and space and time dimensions (Space-Time MOC, STMOC). Figure 1 illustrates the role MOC2.0 plays within the IVOA architecture (Dowler and Evans et al., 2021).

The encoding method described in this document allows one to define and manipulate space and time coverage in such a way that basic operations like union, intersection, equality test can be performed very efficiently. This methodology allows VO applications and data servers to build efficient procedures to perform such operations on observations and catalogs. In the next sections we will describe the different MOCs and their encoding standards.

2 The rationale

The goal behind the MOC is to get a method to manipulate coverages in order to provide very fast union, intersection and equality operations between them. In order to accomplish this task, we based the system on a regular and hierarchical discretization as exposed below. The standard MOC1.0 was limited to space, but for a multitude of use cases in astronomy we need the notion of time to be properly integrated, e. g.:

- What are the space and time coverages of the 2MASS observations and are there any observations which are coincidental with the HST archive?
- Which are the astronomical catalogs which have data for a list of Supernova events within a given time window?
- Are there any other observations coincidental with this gravitational wave detection given its time and spatial coordinates?
- Are there quasi-simultaneous observations (within a given time window) of these two surveys for a list of eclipsing binaries?
- Find the intersection between the SDSS coverage and the ephemeris of this Near Earth Object, was it observed by SDSS? And by Galex? By both missions simultaneously? If so, are there detections within the source catalogues?
- Has Neptune been observed by DSS?

It was possible to answer those questions with MOC1.0 standard and other VO tools but the amount of manipulation and computation was quite a big hurdle for the researchers. With MOC2.0 it is possible to answer these questions in a few milli-seconds. Another example of usage would be the visual inspection of the spatio-temporal coverage of PanSTARRS observations (see Figure 2). The choice of a temporal resolution and a spatial resolution makes it possible to obtain a MOC of a desired data volume.

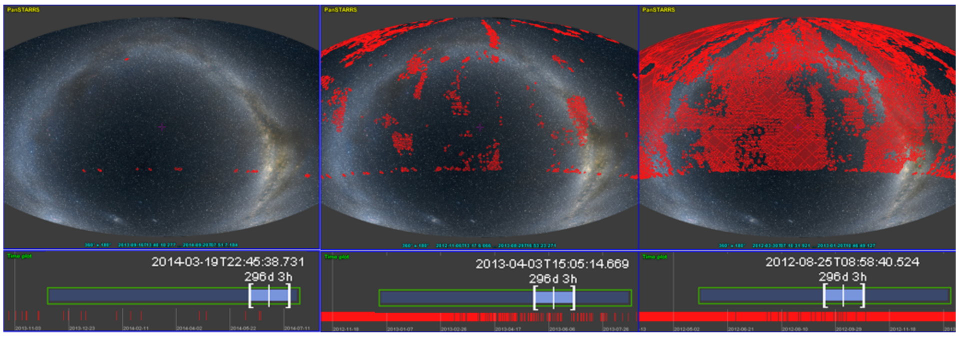

Figure 2: PanSTARRS observations and the associated spatial and temporal coverage within three different periods of time. The volume of the PanSTARRS MOC at a temporal resolution of about 17 minutes and spatial resolution of 52 arcsec is 320 MB.

2.1 Comparing the coverage of multiple data sets

The computation of data set intersections using the MOCs is simple (it is simply a list comparison). The result of any operation is itself a MOC which can be used in further operations. For instance it is possible to compute the intersection of Saturn's ephemeris and the spatio-temporal coverage of HST ACS observations, and subsequently, query a database for retrieving images for which time and position fall within this intersection (see Fig. 3).

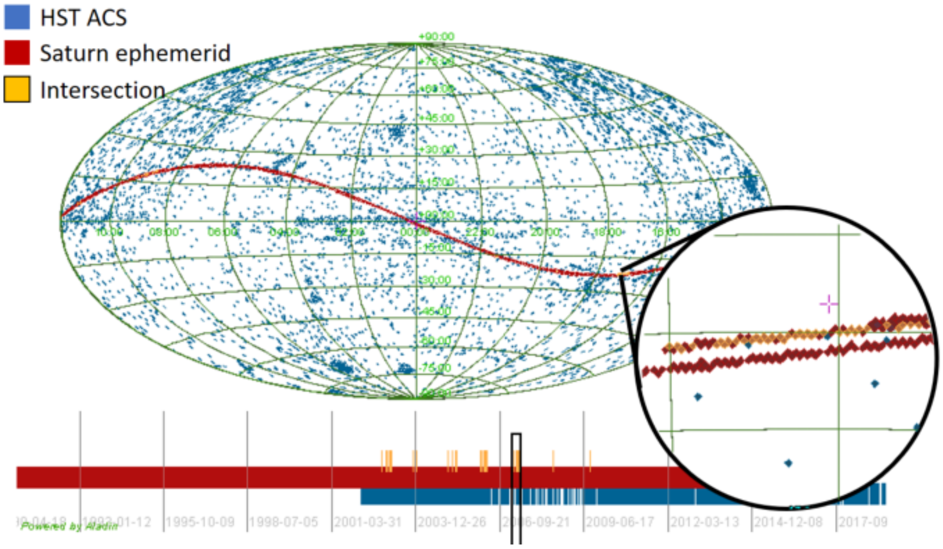

Figure 3: Intersection of HST ACS observations and Saturn ephemeris Space-Time-MOC

2.2 Query databases using MOC

In principle, querying a positional database using complex sky region coverage is possible using ADQL. However, this is rarely possible when the sky region has to be described as unions and intersections of sub-regions to cover a complex, non regular area. In practice, most existing ADQL implementations only support simple regions (cones, boxes, polygons), and can rarely deal with unions and intersections unless by joining independent queries – and this even if the described region is, in fact, empty! If databases are adapted to supporting MOC based queries, they will offer then a useful method allowing any kind of sky region query. In addition, if the internal spatial index of the database is itself based on HEALPix, the filtering will then be straightforward and all the intermediate sky computations will be removed providing an optimal response time.

2.3 Gravitational Wave localisations

The contours of a gravitational-wave sky localization are constructed as follows. The pixels from most probable to least are ranked, and summed up to get a fixed level of probability (Singer and Price et al., 2014). In practice, the HEALPix pixels inside a given contour plot are extracted, and the MOC coverage is generated from the table made up from the pixels. Every single level of probability can be used as a regular MOC even in the case in which the sky localization is irregularly shaped with disjoint regions. This coding technique allows for an extra fast integration in the existing Virtual Observatory structures and tools[1]. The 2D contours of a GW sky localization can be visualised and manipulated using Aladin Desktop, allowing one to compare them with existing surveys, overlap sky map generations with increasing accuracy and computational cost and query the VizieR database. These sets of tasks can also be performed via Python using the astropy affiliated package mocpy[2], efficiently displayed in javascript applications with Aladin Lite, and integrated within Jupyter notebooks through the ipyaladin widget[3].

Adding temporal information in a space MOC encoding can provide a systematic approach in extracting information from follow-up campaigns involving tens of ground and space-based observatories in searching for gravitational-wave counterparts. For illustrative purposes only, a potential use of the time-space MOC is provided when a kilo/macronova emission is a credible counterpart of a gravitational-wave event. Figure 4 shows a mock electromagnetic follow-up campaign of a gravitational-wave sky localization

---

[1] `https://emfollow.docs.ligo.org/userguide/`

[2] `https://cds-astro.github.io/mocpy/`

[3] `https://pos.sissa.it/357/031/pdf`

over a period of time (left panel), a schematic kilonova light-curve and temporal coverage of observations (top right panel) and associated spatial coverage (bottom right panel). This approach permits us to depict the approximate timeline of the instruments involved in the observational campaign and place constraints on the emission properties during the source evolution.

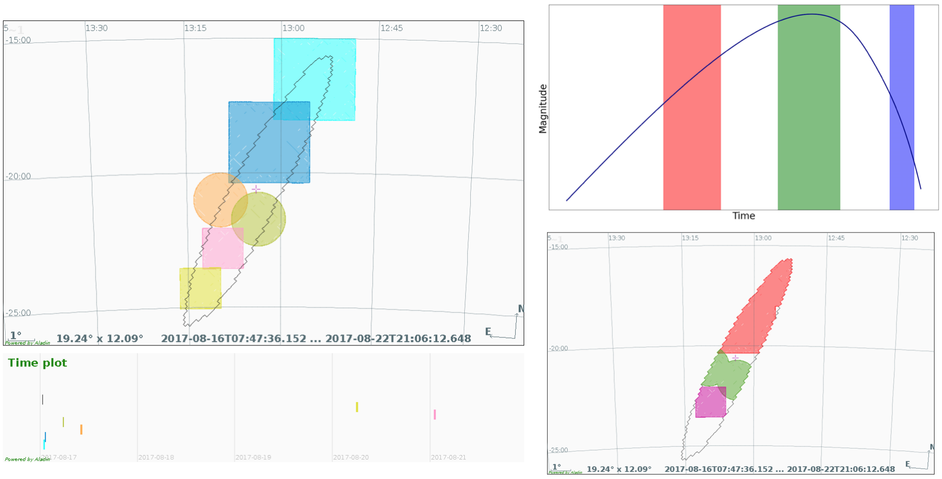

Figure 4: A mock electromagnetic follow-up campaign of a gravitational-wave sky localization over a time period (left). A schematic kilonova light-curve with the observations temporal coverage (top right) and associated spatial coverage (bottom right).

2.4 Space and Time MOC: Einstein Telescope and Early Warning Alerts

The space and time MOC provides us with an effective way to develop new multi-messenger data analysis tools that will have a crucial role when the third-generation interferometric gravitational wave observatories, such as the Einstein Telescope (ET), will begin operation. Here we figure out a few potential applications. ET will explore the universe with gravitational waves up to cosmological distances with an expected detection rate of order $10^5 - 10^6$ black holes and 7×10^4 neutron star mergers per year (Maggiore and Van Den Broeck et al., 2020). For fast and real time data access, the user can query by a specific time range the gravitational-wave sky localizations encoded as a space and time MOC.

In addition, the ET sensitivity at low frequencies enables enough signal-to-noise ratio to accumulate before the merger, making possible a pre-merger gravitational-wave detection and warning for the electromagnetic/neutrino follow up. The simulations show that, by requiring a signal-to-noise ratio >= 12 and a sky localization smaller than 100 deg^2, ET can send an early warning alert between 1 and 20 hours before the merger (with the mean of the distribution at about 5 hours) for signals at 40 Mpc (Chan and Messenger et al., 2018). The electromagnetic/neutrino survey can benefit in multiple spatial and temporal intersections with a gravitational-wave sky localization to probe any electromagnetic/neutrino signals temporally and spatially connected to the inspiral, merger or ring-down phases. Early warning alerts are also planned in the LIGO-Virgo-KAGRA O4 run with an experimental capability to produce and distribute early warning gravitational-wave alerts up to tens of seconds before merger[4].

2.5 Multi-site positional and temporal search

Often, a typical query from a virtual observatory (VO) user is to request all possible records from the VO at a given sky position and/or at a given time. While this is in principle possible by dispatching one narrow positional query to every registered Cone Search, SSA or SIA service and filtering by time, in practice, the number of queries required leads to an unacceptable load on both clients and services. Moreover, most of these queries will deliver no results since most services often lack coverage in the queried region. If basic footprint/coverage information was available for all registered services, for instance using VODataService 1.2 (Demleitner and Plante et al., 2021) only those with coverage in the region and time of interest would then be queried. This would provide a great reduction in the number of services to be queried optimizing the response time. Using the MOC offers the opportunity to provide this coverage information in a uniform way. The MOC could be stored locally for a given service or centrally where the coverage for a number of services would be supported.

3 MOC principles

The MOC standard is defined using four basic building blocks: discretization, unique reference system, hierarchization and efficient encoding:

1. Determine a proper tessellation/discretization methodology for each dimension axis (space, time, ...);
2. Fix a unique referential system for each dimension, to avoid reference conversions and thus allowing to easily compare different data collections;

---

[4] `https://emfollow.docs.ligo.org/userguide/early_warning.html`

3. Use an hierarchical procedure and a unique representation (canonical form) for compacting and quickly manipulating each axis coverage at any level of accuracy;
4. Implement at least one serialization in a binary encoding format (other serializations are possible, e.g. ASCII).

With these principles, a MOC consists of a list of numbers which represent the indices of the cells mapping the coverage of the spatial or temporal axis. As soon as the consecutive cells are used at order n, they will be hierarchically grouped in their parent cell at order n-1, and this recursively. This introduces the notion of orders and associated cell index. The cell boundary alignment implied by the hierarchical structure facilitates the combination of cells at different orders. To work efficiently on existing hardware, we encode of any pair (order, index) as a long integer (64 bits), and we reserve the two most significant bits to encode the type of MOC (spatial, temporal, or future usages). The earlier MOC standard was limited to spatial coverage. We are reusing these principles to manipulate temporal coverages, as well as space-time coverages where we can manipulate the two physical dimensions simultaneously.

We will now explain the conventions chosen for the spatial and temporal axis.

3.1 Space MOC conventions

Defining a sky region by a subset of regular sky tessellation or tiles is not a new idea. In astronomy, one could find three main methods of partitioning the sphere : Q3C, HTM and HEALPix which are respectively using cells in the form of squares, triangles and diamonds for mapping the celestial sphere.

Several publications have compared these methods (O'Mullane and Banday et al., 2001). We justified the choice of HEALPix for the MOC because of these four points:

- Equal areas: by construction, HEALPix consists of diamond cells with equal spherical surfaces. Thus the area of a given region is trivial to compute;
- Computing time: HEALPix has the peculiarity that the computing time does not depend on the hierarchical order [5] (no recursive algorithm);

---

[5] Note that in the HEALPix document orders are refered to as levels, and cells as diamonds.

- Accuracy: HEALPix provides libraries which allow the calculation up to accuracy of 0.4 mas (order 29) (`http://sourceforge.net/projects/healpix/`);
- Standard: Existence of many HEALPix libraries: C++, Java, Fortran, IDL... Also, HEALPix was selected for several all sky missions such as WMAP, Planck and Gaia. The HEALPix main web site is located at Jet Propulsion Lab (`http://healpix.sourceforge.io/`).

The HEALPix (Górski and Hivon et al., 2005) tessellation technique divides the sphere into 12 cells, each of them sub-divided into 4 cells recursively (see Fig. 5). Thus the sphere at order 1 will consist of 48 cells, 192 cells at order 2, 768 at order 3 and so on where each cell at a given order is covering an equal area of the sphere.

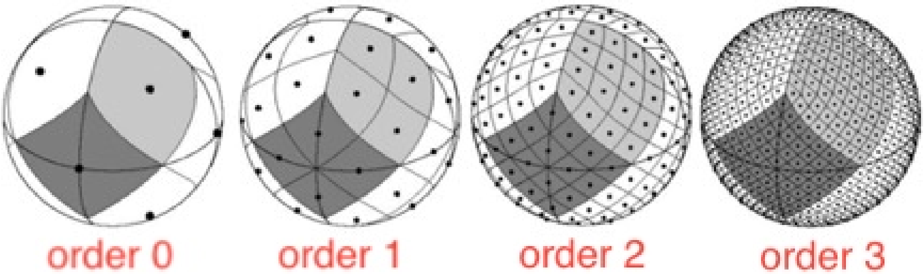

Figure 5: HEALPix partition of the sphere

HEALPix allows three coordinate systems: galactic, equatorial and ecliptic. Allowing various coordinate systems would limit the possibility to compare efficiently SMOCs. There is indeed no equivalence between an HEALPix cell described in a given coordinate system and a cell, or a list of sub-cells expressed in a different coordinate system. Consequently, the SMOC definition is expressed in equatorial coordinate using the ICRS reference. This choice has been motivated by looking at most catalogs and realizing that most of them are using equatorial coordinates.

To support the encoding based on 64-bit longs, the best resolution available is provided at order 29 and according to the HEALPix equations, corresponds approximately to 0.4mas. The SMOC resolution is set by the maximum value of the HEALPix order used to define a region. Its selection depends on the accuracy chosen by the provider to define the region. As data set boundaries are not aligned with the HEALPix cell borders, a SMOC is generally an upper-approximation of the data set coverage. The quality of this approximation depends directly on the chosen SMOC resolution (MOCORD_S). Table 1 provides the HEALPix cell angular resolution for each HEALPix order.

In Figure 6 we show the MOC creation, from images to their coverage, and indicating their corresponding HEALPix numbers.

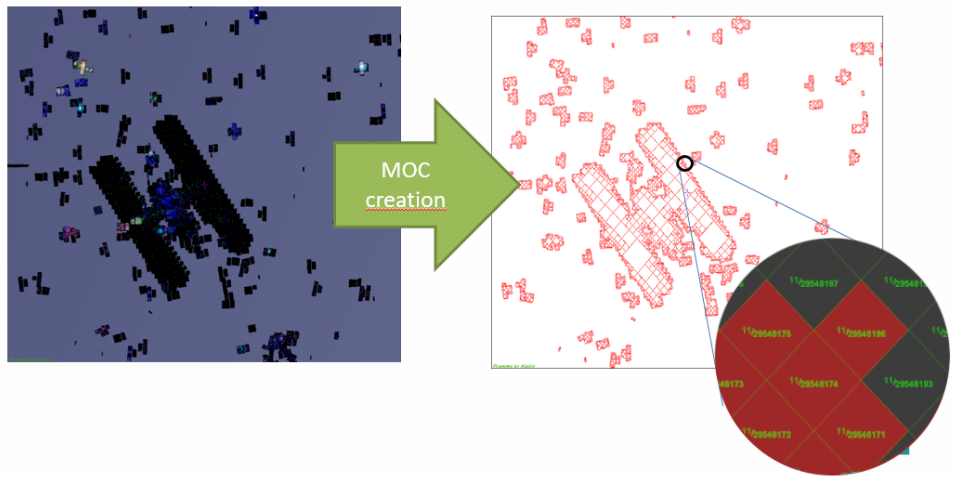

Figure 6: SMOC: from the image to the list of numbers based on HEALPix hierarchy tessellation

Order	Mean Cell Resolution	Order	Mean Cell Resolution
0	58.63°	15	6.442”
1	29.32°	16	3.221”
2	14.66°	17	1.61”
3	7.329°	18	805.2 mas
4	3.665°	19	402.6 mas
5	1.832°	20	201.3 mas
6	54.97′	21	100.6 mas
7	27.48′	22	50.32 mas
8	13.74′	23	25.16 mas
9	6.871′	24	12.58 mas
10	3.435′	25	6.291 mas
11	1.718′	26	3.145 mas
12	51.53”	27	1.573 mas
13	25.77”	28	786.3 μas
14	12.88”	29	393.2 μas

Table 1: SMOC cell resolutions for each order. The SMOC HEALPix cell has constant area, not constant linear dimensions.

Order	Time Cell Resolution (μs)	Order	Time Cell Resolution (μs)
0	2305843009213693952 (≃73117.8y)	31	1073741824 (≃17.9m)
1	1152921504606846976 (≃36558.9y)	32	536870912 (≃9m)
2	576460752303423488 (≃18279.4y)	33	268435456 (≃4.5m)
3	288230376151711744 (≃9139.7y)	34	134217728 (≃2.2m)
4	144115188075855872 (≃4569.9y)	35	67108864 (≃1.1m)
5	72057594037927936 (≃2284.9y)	36	33554432 (≃33s)
6	36028797018963968 (≃1142.5y)	37	16777216 (≃16s)
7	18014398509481984 (≃571.2y)	38	8388608 (≃8s)
8	9007199254740992 (≃285.6y)	39	4194304 (≃4s)
9	4503599627370496 (≃142.8y)	40	2097152 (≃2s)
10	2251799813685248 (≃71.4y)	41	1048576 (≃1s)
11	1125899906842624 (≃35.7y)	42	524288 (≃524ms)
12	562949953421312 (≃17.8y)	43	262144
13	281474976710656 (≃8.9y)	44	131072
14	140737488355328 (≃4.5y)	45	65536
15	70368744177664 (≃2.2y)	46	32768
16	35184372088832 (≃1.1y)	47	16384
17	17592186044416 (≃203.6d)	48	8192
18	8796093022208 (≃101.8d)	49	4096
19	4398046511104 (≃50.9d)	50	2056
20	2199023255552 (≃25.4d)	51	1024
21	1099511627776 (≃12.7d)	52	512
22	549755813888 (≃6.3d)	53	256
23	274877906944 (≃3.2d)	54	128
24	137438953472 (≃1.6d)	55	64
25	68719476736 (≃19.1h)	56	32
26	34359738368 (≃9.5h)	57	16
27	17179869184 (≃4.8h)	58	8
28	8589934592 (≃2.4h)	59	4
29	4294967296 (≃1.5h)	60	2
30	2147483648 (≃35.8m)	61	1

Table 2: TMOC cell resolutions for each orders.

3.2 Time MOC conventions

In order to represent time coverage, we need to select a-priori the total range of time that we will cover with the notation. Following the same SMOC principles, we need to use a discrete time axis where each unit element of this axis has a constant duration. We adopt the Julian Date convention, very common in astronomy and a nominal resolution of 1μs. The temporal dimension being by nature 1D unlike the spatial dimension, we opt for a order progression by factor of 2 (4 for SMOC) and therefore 62 orders (30 for SMOC). This way we can address 2^{62} cells in an unsigned 64-bit integer, i.e. a little bit more than 73000 years at 1μs resolution, enough for most astronomical time events. At the deepest order (61) the TMOC cell number is the number of μs since JD=0.

The time is a relative observation, and depends on the position of the observer. There are many time scales for measuring time: Terrestrial Time (TT), Barycentric Coordinate Time (TCB), Geocentric Coordinate Time (TCG), Ephemeris Time (ET), Barycentric Dynamic Time (TDB), International Atomic Time (AI), etc. We opt for the TCB reference (see Rots and Bunclark et al. (2015) for details). Our choice is motivated by the fact that this system is linear by construction and has been adopted by numerous missions such as Gaia.

It may be necessary to convert the temporal events to the chosen scale. If the ephemeris required for this conversion are not available, opt to degrade the accuracy of the time measurement (typically around 20 minutes for observations within the Earth orbit environment to cover all possible observer positions). Table 2 is showing some time values at a given order.

In Figure 7 we show the creation of TMOC, from a time series to the list of numbers based on time discretization.

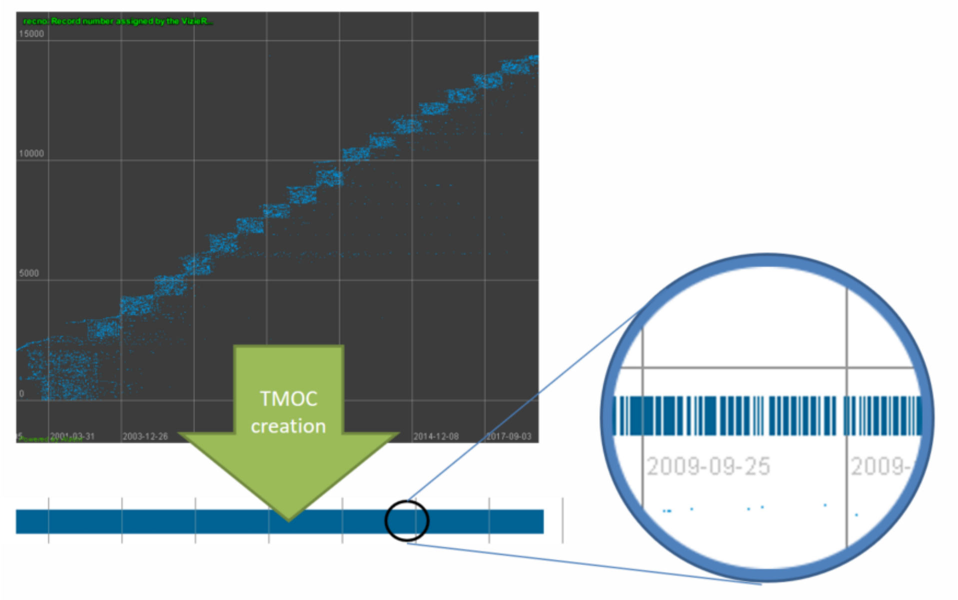

Figure 7: TMOC: from the time series to the list of numbers based on time discretization

3.3 Space and Time MOC conventions

To respond to the different use cases presented at the beginning of the document, the SMOC and TMOC independently are not enough. We need to link the two dimensions in a global mechanism. In other words, we need to be able to select the SMOC using a time window or to select a TMOC using a spatial constraint. Implementing this linkage would allow the potential users to select and interact with the astronomical collections which support space and time and use logical operators between them.

Our approach is to combine these two dimensions - time and space - by associating each time period (coded according to the TMOC convention) with its spatial region (coded according to the SMOC convention). For that, we interleave the information of time coverage with the information of space coverage for this period.

This two-dimensional interleaving approach has the advantage of making the resolutions chosen for time and for space independent. For instance, it is possible to describe observation coverage with a low resolution for time while using a high spatial resolution. A single coding for indexing space and time simultaneously would imply at best very low resolution MOCs due to the 64-bit coding constraint. We thus proposed the interleaving algorithm which allows us to define and manipulate high resolution STMOCs of reasonable sizes for fast algorithms (see Appendix E for STMOC performance). In Figure 8 we show the visual representation of an STMOC, in which at a given TMOC range we obtain the corresponding SMOC.

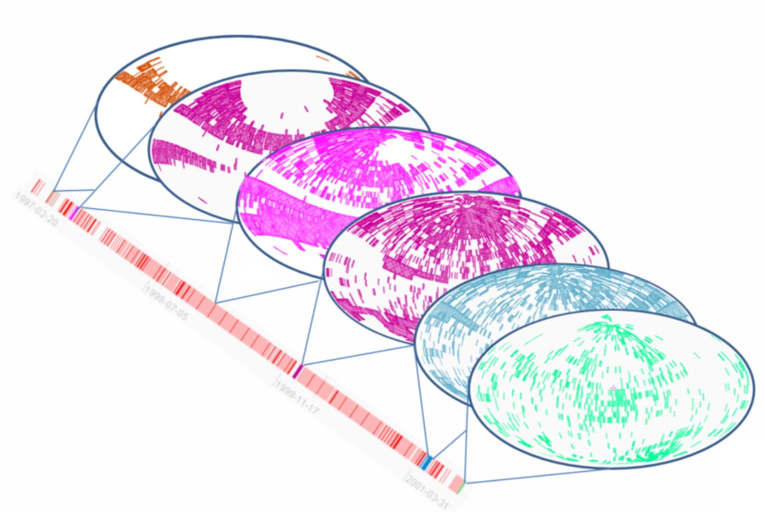

Figure 8: STMOC visual representation in which at a given TMOC range we obtain the corresponding SMOC

4 SMOC and TMOC encoding

The encoding described in this section guarantees backward compatibility with MOCs corresponding to previous versions of this standard.

4.1 Space MOC or SMOC

As introduced above, the SMOC **should** be based on the HEALPix tessellation of the sphere, expressed in the ICRS coordinate reference system for celestial coverages. This document does not describe the use of SMOC outside celestial scope. However, it is possible to use SMOC for other coverages, such as planetary coverages. The definition of the unique reference for each body will have to be defined. Two complementary encoding formats are defined: a string serialization based on ASCII and a binary format based on FITS.

4.1.1 Numbering

The numbering scheme used in SMOC for specifying the cell indices **must** follow the "NESTED" HEALPix numbering schemes (Górski and Hivon et al., 2005). This numbering consists of enumerating all cells in a specific order. For instance, at order 1, there are 48 cells (12x4) enumerated from 0 to 47. In this scheme, the 4 sub-cells of cell M have the indices: $(\mathrm{M}\times4)+3$, $(\mathrm{M}\times4)+1$, $(\mathrm{M}\times4)+2$, $(\mathrm{M}\times4)$ in reading order. And reciprocally, the parent index of cell N is $\mathrm{N}/4$. Each SMOC cell is coded by a pair of numbers: (order, index) which are the HEALPix order and the HEALPix index in this order.

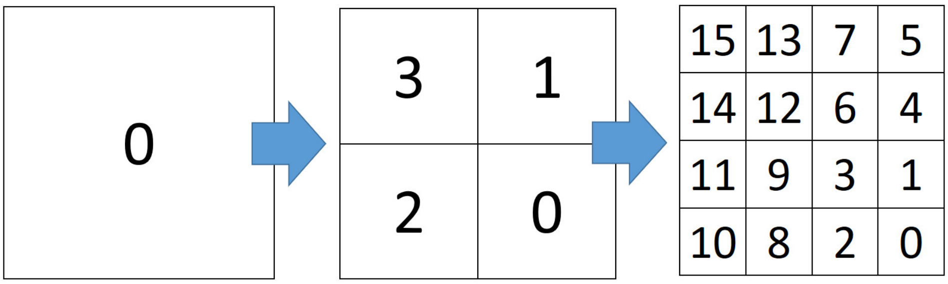

Figure 9: HEALPix numbering principle

Note: The order 0 is a special case, it contains only 12 cells enumerated from 0 to 11.

4.1.2 Sky coordinates

The mechanism used to determine which HEALPix cell contains a given sky location is described in the main article defining the HEALPix system (Górski and Hivon et al., 2005). Several support libraries supporting the most important set of primitives are already available. These libraries are required if one wants to generate SMOCs, and also wants to compare them with sky coordinates. Though please note that these libraries do not have built-in support to performing basic SMOC arithmetics.

4.2 Time MOC or TMOC

As introduced above, TMOC **must** be based on JD system, the time scale TCB, and the Solar System Barycenter as the reference position (see also Fernique and Durand et al., 2019) and section 3.2. The best resolution supported by TMOC is 1μs.

In the case that the time scale and the time reference position are unknown, we recommend to set the time resolution of the generated TMOC to order 31, e.g. about 1000 seconds (see Table 2) corresponding to about twice the light travel time correction between the Earth and the Solar Barycenter. Please refer to the VO note on TIMESYS for more information about this limitation (Demleitner and Bonnarel et al., 2018).

4.2.1 Numbering

The numbering scheme used in TMOC for specifying the time cell indices **must** reuse a similar hierarchical principle as for the SMOC with the difference that the time line has only one dimension, so the hierarchical progression uses a factor of 2 instead of 4, and there is no need to use a HEALPix mapping.

TMOC has 62 orders, and at the best resolution (order 61), a time event will be coded by the integer value representing the number of μs of this event since JD=0.

Two consecutive cells at order N with the indices $(M\times2)+1$, $(M\times2)$ will be coded at order N-1 with the index M and thus recursively. The order N-1 cell duration is 2 times more than the N cell duration (61: 1 μs, 60: 2 μs, 59: 4 μs, etc...).

4.3 Serialization

A MOC can be manipulated and serialized either as a list of cell numbers for each order (hierarchical view), or as a list of intervals at the deepest order (range view). These two methods are used for SMOC, TMOC and STMOC serializations and are presented below.

4.3.1 Binary serialization

To encode a MOC in a FITS file, each MOC pair (order, index) **must** be stored in a FITS binary table. Two packaging modes are defined: either all

MOC pairs (order, index) are stored individually thanks to NUNIQ packaging, or all ranges of indices at the deepest order are stored following the RANGE packaging.

NUNIQ Packaging (valid for SMOC only)

The NUNIQ scheme defines an algorithm for packaging a MOC pair (order, index) into a single integer for compactness:

$$uniq = 4 \cdot (4^{\,order}) + index$$

The inverse operation is:

$$\begin{aligned} order &= \log_2(uniq/4)/2 \\ index &= uniq - 4 \cdot (4^{\,order}) \end{aligned}$$

The list of cells **must** be well-formed (see Section 7.1) allowing to express both hierarchy or range representation. The resulting list is stored in a single-column binary table extension. For orders strictly lower than 14 these UNIQ values can be stored in a 32-bit signed integer (TFORM1='J') , and for the higher orders in a 64-bit signed integer (TFORM1='K').

RANGE packaging

For the coding of RANGE alternative packaging, all the indices are expressed at the maximum resolution, and it is the succession of intervals that will be stored in the FITS table as two 64-bit signed integers (TFORM1='K') : the smallest index of the interval and the index strictly greater than the largest value of the interval. The RANGE values **must** be in ascending numerical order. The resulting list is stored in a single-column binary table extension.

Backward compatibility

RANGE packaging has been introduced for MOC2.0. This method is generally faster than the previous one for reading or writing a MOC because the internal representation of MOC in memory is often range oriented. However, we recommend to use the first method for SMOC for compatibility with existing libraries not yet compatible with MOC2.0.

4.3.2 ASCII serialization

To encode a MOC as a string each MOC pair (order, index) **must** be written sequentially in an ASCII stream as two ASCII numbers separated by slash ("/": decimal ASCII code 47). The order and the slash prefix **may** be omitted if the previous cell has the same order. The elements are separated by one or several space characters (space, CR, LF) corresponding respectively to the decimal ASCII codes: 32, 13, and 10.

The usage of a range operator is allowed in the list of indices using the dash ("-": decimal ASCII code 45) as a separator: lowindex-highindex. The list of cells **must** be well-formed, and the values **must** be in ascending numerical order.

If the best resolution of the MOC (moc order) is greater than the greatest stored order, the moc order **must** be provided, followed by a slash ("/") without any associated index value. In the following example all the cells underneath the explicit pair (order, cell) are implicitly covered up to order 8, the moc order, annotated followed by the terminator "/". Without the terminator we would only have the information of the explicit pair (order, cell), and the assumed best resolution would be at order 2.

Example of an ASCII MOC:

```
1/1 2 4 2/12-14 21 23 25 8/
```

	1/1	2	4	2/12-14	21	23	25	8/
	↓	↓	↓	↓	↓	↓	↓	↓
Order:	1	1	1	2	2	2	2	8
Cell:	1	2	4	12 to 14	21	23	25	

MOC ASCII encoding

EBNF definition of an ASCII MOC:

```
smoc ::= ’s’? moc
tmoc ::= ’t’? moc
stmoc ::= (’t’ moc ’s’ moc)+
moc ::= ordval (sep+ ordval)* [sep+ order]
ordval ::= order sep* vals
order ::= int ’/’
vals ::= val (sep+ val)*
val ::= int | (int ’-’ int)
sep ::= [ \n\r]
int ::= [0-9]+
```

Note that we use Extended BNF supporting regular expression syntax with the following rules: i) postfix * means "repeated 0 or more times”; ii) postfix + means "repeated 1 or more times”; iii) postfix ? means "0 or 1 times". The first three rules depend on the MOC type, i.e. SMOC for space, TMOC for time and STMOC for space-time.

5 STMOC encoding

Coding STMOC consists in the following: for each element of a temporal coverage we list the associated spatial coverage using the natural packaging as defined in the previous section.

5.1 ASCII Serialization

The ASCII serialization of a STMOC is a string following the ASCII MOC serialization presented below, which interleaves time coverage as a excerpt of TMOC and associated space coverage as a SMOC. Each TMOC element **must** be prefixed by the character 't', and each SMOC element **must** be prefixed by the character 's'. The character is thus omitted until the next dimension element is defined.

Example of an ASCII STMOC:

```
t61/1 s29/0-2 t61/3 s28/0 t60/2 61/6 s29/2 5
```

	t61/1	s29/0-2	t61/3	s28/0	t60/2	61/6	s29/2	5
	↓	↓	↓	↓	↓	↓	↓	↓
Dimension:	Time	Space	Time	Space	Time	(Time)	Space	(Space)
Order:	61	29	61	28	60	61	29	(29)
Cell:	1	0 to 2	3	0	2	6	2	5

STMOC ASCII encoding: two independent numbering. Values in parenthesis are implicit from the previous encoding substring.

5.2 Binary Serialization

The binary serialization of a STMOC is a FITS binary table following the RANGE packaging presented previously, which interleaves time range(s) and their corresponding space coverage ranges. Following the binary RANGE serialization method described below, each range (time or space) is coded as two 64-bit signed integers ([min..max[). To distinguish time and space indices, the time indices **must** have the 64th bits forced to 1. It is not a sign inversion (two's complement) but a mask affecting only that last bit without touching any other bits. The order of dimensions is always time first.

Illustration of STMOC interleaving method

This list of ranges will be coded in a list of 64bits integers (time indices with the 64th bit forced to 1) as:

```
tmin1 tmax1 smin1 smax1 tmin2 tmax2 tmin3 tmax3 smin2 smax2 smin3 smax3
```

STMOC encoding must conform to the following simple rules:

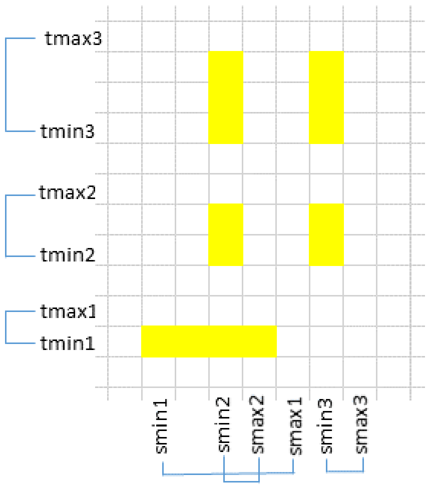

Figure 10: STMOC encoding with two independent numbering system

- Temporal cells which are sequential and have the SAME spatial coverage MUST be aggregated in the coding scheme.
- The cell order MUST also be increasing first on the temporal axis, then on the spatial axis.

This is illustrated in Figure 10.

6 FITS keywords

For the binary representations which are packaged in binary FITS table, we define a set of FITS keywords, their possible values and set when those fields are required, optional or recommended in Table 3 and show an example of FITS headers for a MOC. Since MOC 1.1 (Fernique and Boch et al., 2019) the `PIXTYPE = "HEALPIX"` keyword/value is no longer required, and should be omitted. The keyword `MOCORDER` is no longer required either, but it can be used for backwards compatibility if required. Other FITS Keywords could be used to augment the information like DATES and ORIGIN.

Example of FITS headers for a MOC:

```
SIMPLE  =                    T
BITPIX  =                    8
NAXIS   =                    0
EXTEND  =                    T
END
```

Keyword	**Definition**	**MOC1.1**	**MOC2.0**
MOCVERS	The version of the MOC encoding standard. If it is following this document (TMOC, STMOC and STMOC), it **must** be '2.0'. If not defined it is assumed to be 1.1.	NA	mandatory
MOCDIM	Physical(s) dimension(s). Either 'SPACE' for SMOC, 'TIME' for TMOC or 'TIME.SPACE' for STMOC. If omitted, 'SPACE' is assumed for backward compatibility.	NA	mandatory
ORDERING	The packaging method used. It is either NUNIQ (V1.1 or V2.0 SMOC) or RANGE (V2.0).	mandatory	mandatory
COORDSYS	The coordinate system in use. The value **must** be 'C' for SMOC (ICRS).	mandatory	mandatory
TIMESYS	The time system in use. The value **must** be 'TCB' for TMOC.	NA	mandatory
MOCID	Original data identifier. For MOCs that are coverages of VO resources, in particulary those used in VODataService 1.2 coverage elements (Demleitner and Plante et al., 2021), MOCID can contain the IVOA id of the VO resource described.	optional	optional
MOCTOOL	The name of the MOC software generator. It is also recommended to add the software version number to its name.	optional	optional
MOCTYPE	Provenance data type. Either 'IMAGE', or 'CATALOG'. In the first case for areas computed from existing images and/or footprint or even STC strings, in the second case for areas computed from a collection of point sources using a unique and/or derived area.	optional	optional
MOCORD_S	Best resolution of the space dimension, expressed as the order.	NA	mandatory
MOCORD_T	Best resolution of the time dimension, expressed as the order.	NA	mandatory
MOCORDER	Best resolution of the space dimension, expressed as the order.	mandatory	NA
PIXTYPE	'HEALPIX'	mandatory	NA

Table 3: FITS Keywords for MOC.

```
XTENSION = ’BINTABLE’         / HEALPix Multi Order Coverage map
BITPIX  =                    8
NAXIS   =                    2
NAXIS1  =                    4
NAXIS2  =                16461
PCOUNT  =                    0
GCOUNT  =                    1
TFIELDS =                    1
TFORM1  = ’1J     ’
TTYPE1  = ’UNIQ   ’           / HEALPix UNIQ pixel number
ORDERING = ’NUNIQ ’           / NUNIQ coding method
COORDSYS = ’C     ’           / ICRS reference frame
MOCDIM  = ’SPACE  ’           / Physical dimension
MOCORD_S =                 12 / MOC resolution (best order)
MOCTOOL = ’Aladin11.1’        / Name of the MOC generator
MOCTYPE = ’CATALOG ’          / Source type (IMAGE or CATALOG)
MOCID   = ’ivo://CDS/I/259’   / Identifier of the collection
MOCVERS = ’2.0     ’          / MOC standard version
ORIGIN  = ’ivo://CDS’         / MOC origin
DATE    = ’2013-06-15T11:50:43’ / MOC creation date
EXTNAME = ’Tycho MOC’         / MOC name
END
```

7 MOC usage constraints

7.1 Canonical form

The speed of MOC operations - creation, union, intersection, etc is directly dependent on the speed of the equality test. It is therefore essential to always express a MOC in a canonical way, ie one unique representation for one coverage. Thus in the case of a hierarchical representation a MOC **must** be "well-formed", i.e. redundant cells are not allowed, the cells must be ascending sorted and the hierarchical encoding principle must be respected. Thus it is not allowed to encode sibling cells instead of their parent (4 siblings for SMOC, 2 siblings for TMOC). In the case of range representation, the list of ranges must be expressed without overlapping and sorted ascending.

7.2 Compromise of Volume VS. Resolution

In order to easily handle MOCs, it is recommended to adjust the maximum resolution, i.e. the deepest order, to obtain a representation of the desired data volume even if it means degrading the accuracy of the coverage (see Appendix E). In the case of STMOC, it is possible to adjust the spatial order and/or on the temporal order independently.

7.3 Working resolution

The MOC has been designed to be able to efficiently handle observation coverages (images, catalogs, ...). During the construction of the MOC, we must then ensure that at the chosen nominal resolution, any cell of the MOC contains at least one observation (no empty cell). To keep this assumption, during operations (unions, intersections ...) between 2 or more MOCs (e.g: MocA ∪ MocB ∪ MocC) of different resolutions, the operations must always be done at the worst (lowest) resolution of the original MOCs in order not to lose any observations, nor to create empty cells (see Figure 11), and finally to guarantee the set logical properties (commutativity, associativity,...).

Figure 11: Visualisation of the principles behind MOC operators

Note that MOC usage can be diverted to also manipulate surfaces unrelated to observations. When it is used this way such operations (oversampling, surface dilatation or surface erosion) can be applied. And in this context it is necessary to work at the best (highest) resolution of the involved MOCs for preserving the properties of the surfaces operations (commutativity, associativity, ...).

8 Summary and conclusion

We have reviewed the standards for encoding the different MOC flavors, the space MOC (already described in version 1.1 of this document), SMOC, the time MOC, TMOC and the space-time MOC, STMOC. The conventions for space and time MOC are the following:

- The SMOC is simply defined as a list of HEALPix indices (order, index).
- According to HEALPix, the sphere is divided in cells, hierarchically grouped 4 by 4 with 30 orders and the space coverage for the deepest order is approximatively 0.4mas.
- The space reference system is ICRS.
- The TMOC is simply a list of time interval indices (order, index).
- The time scale is divided in intervals hierarchically grouped 2 by 2 with 62 orders and the time coverage for the deepest order is 1 μs.
- The time values are defined using JD $= 0$ as the origin, in Barycentric system.

Once defined and encoded for a given astronomical collection, one can easily combine the MOCs for these two dimensions to create a merged STMOC which can be used to navigate and access the collection through their coverage for both time and space simultaneously. The possibilities are then very interesting and will be a very valuable astronomical tool.

References

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

Chan, M. L., Messenger, C., Heng, I. S. and Hendry, M. (2018), ‘Binary neutron star mergers and third generation detectors: Localization and early warning’, *Physical Review D* **97**(12), 123014, arXiv:1803.09680.
`http://doi.org/10.1103/PhysRevD.97.123014`

Demleitner, M. (2020), ‘Space-Time Coverage in the VO Registry’, *arXiv e-prints* p. arXiv:2007.07519, arXiv:2007.07519.
`https://ui.adsabs.harvard.edu/abs/2020arXiv200707519D`

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.
`https://ui.adsabs.harvard.edu/abs/2021ivoa.spec.1102D`

Demleitner, M.and Nebot, A., Bonnarel, F., Michel, L., Fernique, P. and Boch, T. (2018), 'A Proposal for a TIMESYS Element in VOTable', IVOA Note 29 October 2018.
http://ivoa.net/documents/Notes/TimeSys/

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

Fernique, P., Boch, T., Donaldson, T., Durand, D., O'Mullane, W., Reinecke, M. and Taylor, M. (2019), 'MOC - HEALPix Multi-Order Coverage map Version 1.1', IVOA Recommendation 07 October 2019.
https://ui.adsabs.harvard.edu/abs/2019ivoa.spec.1007F

Fernique, P., Durand, D. and Nebot, A. (2019), Time in Aladin, *in* P. J. Teuben, M. W. Pound, B. A. Thomas and E. M. Warner, eds, 'Astronomical Data Analysis Software and Systems XXVII', Vol. 523 of *Astronomical Society of the Pacific Conference Series*, p. 497.
https://ui.adsabs.harvard.edu/abs/2019ASPC..523..497F

Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M. and Bartelmann, M. (2005), 'HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere', *The Astrophysical Journal* **622**(2), 759–771, arXiv:astro-ph/0409513.
http://doi.org/10.1086/427976

Maggiore, M., Van Den Broeck, C., Bartolo, N., Belgacem, E., Bertacca, D., Bizouard, M. A., Branchesi, M., Clesse, S., Foffa, S., García-Bellido, J., Grimm, S., Harms, J., Hinderer, T., Matarrese, S., Palomba, C., Peloso, M., Ricciardone, A. and Sakellariadou, M. (2020), 'Science case for the Einstein telescope', *Journal of Cosmology and Astroparticle Physics* **2020**(3), 050, arXiv:1912.02622.
http://doi.org/10.1088/1475-7516/2020/03/050

O'Mullane, W., Banday, A. J., Górski, K. M., Kunszt, P. and Szalay, A. S. (2001), Splitting the Sky - HTM and HEALPix, *in* A. J. Banday, S. Zaroubi and M. Bartelmann, eds, 'Mining the Sky', p. 638.
http://doi.org/10.1007/10849171_84

Rots, A. H., Bunclark, P. S., Calabretta, M. R., Allen, S. L., Manchester, R. N. and Thompson, W. T. (2015), 'Representations of time coordinates in FITS. Time and relative dimension in space', *Astronomy & Astrophysics* **574**, A36, arXiv:1409.7583.
http://doi.org/10.1051/0004-6361/201424653

Singer, L. P., Price, L. R., Farr, B., Urban, A. L., Pankow, C., Vitale, S., Veitch, J., Farr, W. M., Hanna, C., Cannon, K., Downes, T., Graff, P., Haster, C.-J., Mandel, I., Sidery, T. and Vecchio, A. (2014), 'The First Two Years of Electromagnetic Follow-up with Advanced LIGO and Virgo', *The Astrophysical Journal* **795**(2), 105, arXiv:1404.5623.
`http://doi.org/10.1088/0004-637X/795/2/105`

A Version History

A.1 Changes between versions 1.1 and 2.0

The differences between version 2.0 of MOC and the preceding version 1.1 are:

- The adaptation of the previous MOC (spatial) to a temporal dimension;
- The definition of the concept of MOC allowing to handle both spatial and temporal MOCs;
- The extension of ASCII and binary coding to support these new concepts.
- Relax the language to allow a future use of MOC with a non-sky coordinate system.

Taking these extensions into account required a major restructuring of the document.

A.2 Changes between versions 1.0 and 1.1

The differences between version 1.1 of MOC and the preceding version 1.0 are:

- The String MOC serialization was moved from an informative section (suggested syntax) to the normative section;
- A MOCORDER convention for String SMOC and JSON SMOC was added. (Demleitner, 2020).

B Suggested algorithms for basic operations

Mapping a MOC to a unique sorted list of cells at the deepest resolution order allows usage of very easy and very fast algorithms. Basic operations such as unions or intersections can be computed via bit shifts and simple dichotomic algorithms on sorted lists. To reduce as much as possible the memory requirement, a good practice is to store range sets of continuous cells [minValue .. maxValue[, instead of individual cells.

B.1 Union: moc1 ∪ moc2

```
Map moc1 to rangeList
Map moc2 in the same rangeList
Unmap the resulting rangeList
```

B.2 Intersection: moc1 ∩ moc2

```
Map moc1 in rangeList1
foreach order/index of moc2
     shift=2*(maxOrder-order)
     append in a rangeList2 the intersection between
           [index << shift .. index+1 << shift[
     and  the corresponding range(s) of rangeList1
Unmap rangeList2
```

B.3 Map: moc To rangeList

```
foreach order/index of moc
      shift=2*(maxOrder-order)
      append in rangeList [index << shift , (index+1) << shift[
      (the range overlapping must be adjusted)
```

B.4 Unmap: rangeList To moc

```
for order = 0 to maxOrder
   end if rangeList is empty
   shift = 2*(maxOrder-order)
   offset = (1<<shift) -1
   foreach range [min..max[ of rangeList
      append in moc order/index where index is in [m1 .. m2[
            m1 = (min+offset) >> shift
            m2 = max >> shift
      remove from rangeList [m1<<shift .. m2<<shift[
```

C Basic HEALPix functions

For generating space MOC from observations, or drawing them on the sphere, an HEALPix library is required. The basic functions available in all HEALPix libraries are the following :

- npix <= coordToNpix(order, alpha,delta) : returns the HEALPix cell index containing the alpha,delta coordinates.
- ArrayOfNpix <= queryDisc(order, alpha,delta,radius) : returns the list of cell indices covering the (long,lat,radius) cone
- ArrayOfNpix <= queryPolygon(order, alpha1,delta1, ⋯ alphaN,deltaN): returns the list of cell indices covering the spherical polygon
- (alpha,delta) <= NpixToCoord(order,npix) : returns the coordinates of the center of order/npix cell.
- ArrayOf(alpha,delta) <= NpixToCorners(order,npix) : returns the corner coordinates of order/npix cell.

D Basic time functions

For generating TMOC from observations, a time library might be required to convert the dates are not expressed in JD TCB

- `http://www.iausofa.org/`
- `http://javastro.github.io/jsofa/`
- `https://docs.astropy.org/en/stable/time/#module-astropy.time`
- Obtain the index from the JD: calculating the TMOC index from a JD expressed as a double can be done by simple multiplication, but will only allow millisecond accuracy around the present time because of the conversion from double to long.

  ```
  long getMicrosec(double jd) {
      return (long)(jd*86400000000L);
  }
  ```

  To guarantee microsecond accuracy, a solution may be to use a second parameter to indicate an origin expressed as a long integer from JD=0, close to the observation dates, and to use the offsets of these observations, expressed as double, from this new origin.

  ```
  long getMicrosec(double offset, long origin) {
      long x = (long)(offset*86400000000L);
      return x + (origin*86400000000L);
  }
  ```

E MOC Volume and Performances

The MOC describes ranges of space and time as an explicit list of cells. The volume can vary a lot from a few bytes to several megabytes. Since MOC is hierarchical, its volume mainly depends on three factors:

- The chosen STMOC resolutions (spatial and temporal).
- The geometry of the region and its time coverage.
- The density of sources for catalogs.

Some examples are shown in Tables 4, 5, 6 and 7.

Order	**Resolution**	**Volume (KB)**	**Generation time (ms)**
6	54.87’	23	36
8	13.74’	33	40
10	3.435’	46	44
12	51.53’	71	49
14	12,88"	215	64
16	3.221"	317	78
18	805.2mas	488	105
20	201.3mas	655	132

Table 4: SMOCs for HST ACS science observations obtained from CADC’s OBSCORE using field central position and not the original footprint.

Order	**Resolution**	**Volume**	**Generation time (ms)**
15	2y 83d	<1KB	18
19	50d 21h	1KB	30
23	3d 4h	7KB	32
27	4h 46m	70KB	34
31	17m 53.7s	673KB	36
35	1m 7.1s	1MB	38
39	4.19s	1MB	38
43	262.1ms	1MB	38

Table 5: TMOCs for HST ACS science observations obtained from CADC’s OBSCORE using the epoch of the observations.

T Order	**S order**	**Volume**	**Generation time (s)**
15	6	269KB	0.3
19	8	511KB	0.3
23	10	867KB	0.5
27	12	1.4MB	2.1
31	14	2MB	4.7
35	16	4.3MB	9.7
39	18	4.3MB	9.8
43	20	4.3MB	10
28	8	1MB	2
19	12	929KB	0.5

Table 6: STMOCs for HST ACS science observations obtained from CADC's OBSCORE using the epoch of the observations and the field central positions.

Operand and Order	**Union**	**Intersection**
SMOC 10	2ms	2ms
TMOC 23	<1ms	<1ms
STMOC 10,23	2ms	7ms
SMOC 12	2ms	1ms
TMOC 27	<1ms	<1ms
STMOC 12,27	6ms	4ms

Table 7: STMOCs operations between HST ACS (211453 observations) and HST WFC3 (276175 observations) science observations obtained from CADC's OBSCORE using the epoch of the observations and the field central positions.

F JSON encoding

If it is required to write a MOC as an JSON string, it is suggested to use the following syntax:

```
{ "order":[index,index,...], "order":[index, index...], ... }
```

As for the ASCII MOC serialization, if the best resolution of the MOC (MOCORDER) is greater than the greatest stored order, the MOCORDER should be provided with an empty index list.

Example of a JSON SMOC or TMOC:

```
{"1":[1,2,4], "2":[12,13,14,21,23,25], "8":[]}
```

As with ASCII encoding, the differentiation of a spatial MOC from a temporal MOC could be done by prefixing the JSON MOC with an 's' or a

’t’ using a dedicated JSON hierarchy level. In the absence of this information, the nature of the MOC is determined by its context of use.

```
{"t": {
  "order":[index,index,...],
  "order":[index, index...], ...
} }
```

or

```
{"s": {
  "order":[index,index,...],
  "order":[index, index...], ...
} }
```

The coding of an STMOC will then be a list of couples (SMOC,TMOC) formalized in the following way:

```
[
   {"t": {
    "order":[index,index,...],
    "order":[index, index...], ... } },
   {"s": {
    "order":[index,index,...],
    "order":[index, index...], ... } },
   ...
   {"t": {
    "order":[index,index,...],
    "order":[index, index...], ... } },
   {"s": {
    "order":[index,index,...],
    "order":[index, index...], ... } }
]
```

If the spatial or temporal orders of the highest `"order":[index,index,...]` pair is lower than the respective spatial or temporal MOCORDER, then add an additional pair at the highest order with an empty index list.

Example of a JSON STMOC:

```
   [  { "t":{ "61":[0]}, "s":{ "29":[0,1,2]},
      { "t":{ "61":[2]}, "s":{ "28":[0]} },
      { "t":{ "61":[]}, "s":{ "29":[]} ]
```