11 Application program interface

11.1 Overview

This clause specifies an API for the operations and concepts defined in this document. It specifies:

a) data types,

b) classes and their methods, and

c) functions.

The data types specified in this clause are composed of basic and structured data types. Data types supporting methods and functions are defined in 11.2. To support the conformance of exchange formats (see 14.3), additional data types for storage and/or transmission are defined in 11.5.

Class specifications serve to organize methods related to specific SRM concepts. In this sense, *class instances* represent SRM concept instances. An API object is an instance of a class. A *class* defines *methods* that produce outputs by operating on the *state* of an object and its inputs. Classes and their methods are defined in 11.3.

Functions are specified outside of the class specifications and operate only on the specified inputs to produce their corresponding outputs. The capabilities provided by these functions include creating instances of standard and set-based SRFs, and querying the extent of support of an API implementation. These functions are specified in 11.4.

Implementations of classes and their methods, and other functions, shall conform to computational accuracy requirements (see 14.2). The computational accuracy requirements do not apply to a sequence or chain of SRF operations, only to each individual SRF operation in the sequence.

Due to variations in class-specific equivalent data representations, it is possible that `Get` methods will return values that are different from previously input parameter values in corresponding `Create` methods (see 11.3.11).

Functional implementation and exchange format conformance are based on profiles (see Clause 12). Conformance of an application to a profile is defined in 14.5.

11.2 Data types

11.2.1 Overview

Data types are organized into *basic data types* and *structured data types*. Basic data types consist of single values, whereas structured data types consist of multiple values. Basic data types include numeric data types, enumerated data types, and selection data types. Selection data types are similar to enumerated data types but can be extended via registration. Structured data types include array data types, record data types and variant record data types. The elements of arrays are all of the same data type and are referenced by position within the array, whereas the elements of records may be of different data types and are referenced by name. In variant records, a selector is used to choose one record data type from among several alternative record data types.

11.2.2 SRFT abbreviated forms

Table 11.1 lists the SRFT names and their abbreviated forms used in the formation of enumerant names and record element names of data types.

Table 11.1 — SRFT abbreviated forms

Abbreviated form	SRFT name
CC	Celestiocentric
CD	Celestiodetic
CM	Celestiomagnetic
EC	Equidistant Cylindrical
EI	Equatorial Inertial
HAEC	Heliospheric Aries Ecliptic
HEEC	Heliospheric Earth Ecliptic
HEEQ	Heliospheric Earth Equatorial
LCC	Lambert Conformal Conic
LCE_3D	Lococentric Euclidean 3D
LSA_2D	Local Space Azimuthal 2D
LSP_2D	Local Space Polar 2D
LSR_2D	Local Space Rectangular 2D
LSR_3D	Local Space Rectangular 3D
LTSAS	Local Tangent Space Azimuthal Spherical
LTSC	Local Tangent Space Cylindrical
LTSE	Local Tangent Space Euclidean
OMS	Oblique Mercator Spherical
PD	Planetodetic
PS	Polar Stereographic
SEC	Solar Ecliptic
SEQ	Solar Equatorial
SMD	Solar Magnetic Dipole
SME	Solar Magnetic Ecliptic
TM	Transverse Mercator

11.2.3 Numbers

Two categories of numbers are specified: integer numbers and floating-point numbers. The general-purpose integer data types are `Integer`, `Integer_Positive` and `Integer_Unsigned`. All implementations that conform to this standard shall support at least the minimum ranges for values of these data types as specified in Table 11.2.

Table 11.2 — Integer data types

Data type	Value range
`Integer`	[−2 147 483 647, 2 147 483 647]
`Integer_Positive`	[1, 4 294 967 295]
`Integer_Unsigned`	[0, 4 294 967 295]

`Long_Float` is a data type defined for floating-point numbers. This data type corresponds to the double precision floating-point data type specified by ISO/IEC/IEEE 60559. However, implementations on architectures that support other floating-point representations are allowed. When recording a `Long_Float` number in a file or archive, the floating-point data type specified in ISO/IEC/IEEE 60559 shall be used. It is the responsibility of the implementation to make suitable conversions when the internal floating-point format differs from the standard floating point data type.

11.2.4 Logicals

The general-purpose logical data type is Boolean. All implementations that conform to this standard shall support this data type as specified in Table 11.3.

Table 11.3 — Logical data type

Data type	Values
`Boolean`	[false (or 0), true (or 1)]

11.2.5 Object_Reference

An *object reference* is a generic reference to a class instance. `Object_Reference` is an opaque data type that implements this concept. If the values of two `Object_Reference`s are equal, they shall refer to the same class instance. In all the method specifications in this clause, whenever an argument passed to or returned from a method is a class instance, it is an `Object_Reference` that is passed or returned.

The `NULL_Object` is defined as a special `Object_Reference`. If the value of an `Object_Reference` is equal to the value of the `NULL_Object`, it does not reference any class instance. On an error condition, some language bindings may require method and/or function outputs to be defined. In these cases, `Object_Reference` outputs shall be set to `NULL_Object` as appropriate.

11.2.6 Enumerated data types

11.2.6.1 Overview

Enumerated data types are data types whose values are specified from an ordered list of names. Enumerated data types are a closed list, the members of which do not change based on registration or deprecation.

11.2.6.2 Axis_Direction_2D

This data type represents the values of the axis direction parameter(s) of the SRFT LOCAL_SPACE_RECTANGULAR_2D.

```
Axis_Direction_2D ::= (   POSITIVE_FIRST_BASIS_AXIS_2D,
                          POSITIVE_SECOND_BASIS_AXIS_2D,
                          NEGATIVE_FIRST_BASIS_AXIS_2D,
                          NEGATIVE_SECOND_BASIS_AXIS_2D)
```

POSITIVE_FIRST_BASIS_AXIS_2D indicates that the primary axis of the LOCAL_SPACE_RECTANGULAR_2D SRF is to be aligned with the positive first basis axis of its LOCOCENTRIC_EUCLIDEAN_2D CS.

POSITIVE_SECOND_BASIS_2D indicates that the primary axis of the LOCAL_SPACE_RECTANGULAR_2D SRF is to be aligned with the positive second basis axis of its LOCOCENTRIC_EUCLIDEAN_2D CS.

NEGATIVE_FIRST_BASIS_AXIS_2D indicates that the primary axis of the LOCAL_SPACE_RECTANGULAR_2D SRF is to be aligned with the negative first basis axis of its LOCOCENTRIC_EUCLIDEAN_2D CS.

NEGATIVE_SECOND_BASIS_AXIS_2D indicates that the primary axis of the LOCAL_SPACE_RECTANGULAR_2D SRF is to be aligned with the negative second basis axis of its LOCOCENTRIC_EUCLIDEAN_2D CS.

11.2.6.3 Axis_Direction_3D

This data type represents the values of the axis direction parameter(s) of the SRFT LOCAL_SPACE_RECTANGULAR_3D.

```
Axis_Direction_3D ::= (   POSITIVE_FIRST_BASIS_AXIS_3D,
                          POSITIVE_SECOND_BASIS_AXIS_3D,
                          POSITIVE_THIRD_BASIS_AXIS_3D,
                          NEGATIVE_FIRST_BASIS_AXIS_3D,
                          NEGATIVE_SECOND_BASIS_AXIS_3D,
                          NEGATIVE_THIRD_BASIS_AXIS_3D )
```


POSITIVE_FIRST_BASIS_AXIS_3D indicates that the specified (primary or secondary) axis of the LOCAL_SPACE_RECTANGULAR_3D SRF is to be aligned with the positive first basis axis of its LOCOCENTRIC_EUCLIDEAN_3D CS.

POSITIVE_SECOND_BASIS_AXIS_3D indicates that the specified (primary or secondary) axis of the LOCAL_SPACE_RECTANGULAR_3D SRF is to be aligned with the positive second basis axis of its LOCOCENTRIC_EUCLIDEAN_3D CS.

POSITIVE_THIRD_BASIS_AXIS_3D indicates that the specified (primary or secondary) axis of the LOCAL_SPACE_RECTANGULAR_3D SRF is to be aligned with the positive third basis axis of its LOCOCENTRIC_EUCLIDEAN_3D CS.

NEGATIVE_FIRST_BASIS_AXIS_3D indicates that the specified (primary or secondary) axis of the LOCAL_SPACE_RECTANGULAR_3D SRF is to be aligned with the negative first basis axis of its LOCOCENTRIC_EUCLIDEAN_3D CS.

NEGATIVE_SECOND_BASIS_AXIS_3D indicates that the specified (primary or secondary) axis of the LOCAL_SPACE_RECTANGULAR_3D SRF is to be aligned with the negative second basis axis of its LOCOCENTRIC_EUCLIDEAN_3D CS.

NEGATIVE_THIRD_BASIS_AXIS_3D indicates that the specified (primary or secondary) axis of the LOCAL_SPACE_RECTANGULAR_3D SRF is to be aligned with the negative third basis axis of its LOCOCENTRIC_EUCLIDEAN_3D CS.

11.2.6.4 Interval_Type

This data type is used to specify coordinate-component intervals.

```
Interval_Type::= (   OPEN_INTERVAL,
                     GE_LT_INTERVAL,
                     GT_LE_INTERVAL,
                     CLOSED_INTERVAL,
                     GT_SEMI_INTERVAL,
                     GE_SEMI_INTERVAL,
                     LT_SEMI_INTERVAL,
                     LE_SEMI_INTERVAL,
                     UNBOUNDED )
```

OPEN_INTERVAL denotes the bounded open interval (a, b).

GE_LT_INTERVAL denotes the bounded interval $[a, b)$.

GT_LE_INTERVAL denotes the bounded interval $(a, b]$.

CLOSED_INTERVAL denotes the bounded interval $[a, b]$.

GT_SEMI_INTERVAL denotes the unbounded interval $(a, +\infty)$.

GE_SEMI_INTERVAL denotes the unbounded interval $[a, +\infty)$.

LT_SEMI_INTERVAL denotes the unbounded interval $(-\infty, b)$.

LE_SEMI_INTERVAL denotes the unbounded interval $(-\infty, b]$.

UNBOUNDED denotes all values $(-\infty, +\infty)$.

For angular values, the terms "$+\infty$" and "$-\infty$" denote the most extreme valid values for the coordinate-component.

EXAMPLE 1 In the latitude coordinate-component interval of type GE_SEMI_INTERVAL with value $[0.0, +\infty)$, "$+\infty$" denotes $+\pi/2$ radians.

EXAMPLE 2 In the longitude coordinate-component interval of type LT_SEMI_INTERVAL with value $(-\infty, 0.0)$, "$-\infty$" denotes $-\pi$ radians.

11.2.6.5 Polar_Aspect

This data type represents the values of the polar aspect parameter of SRFT POLAR_STEREOGRAPHIC.

```
Polar_Aspect ::= (   NORTH,
                     SOUTH )
```

11.2.6.6 SRF_Region_Status

This data type represents coordinate location with respect to the applicable region and/or extended region

(see 8.3.2.4) of an SRF. For backward compatibility, the enumerated values below use "SRF_REGION" as equivalent to applicable region.

```
SRF_Region_Status ::= (   IN_SRF_REGION,
                          IN_EXTENDED_REGION_OUTSIDE_SRF_REGION,
                          OUTSIDE_BOTH_SRF_REGION_AND_EXTENDED_REGION )
```

IN_SRF_REGION denotes a coordinate that is contained within the applicable region.

IN_EXTENDED_REGION_OUTSIDE_SRF_REGION denotes a coordinate that is contained within the extended region, but is not contained within the applicable region.

OUTSIDE_BOTH_SRF_REGION_AND_EXTENDED_REGION denotes a coordinate that is contained within the CS domain, but is not contained within either the applicable region or the extended region.

11.2.6.7 SRF_Region_Type

This data type is used to indicate whether an applicable region or extended region is represented in terms of the coordinate system of the SRF or in terms of the geodetic coordinate system of the Celestiodetic SRF for that ORM.

```
SRF_Region_Type ::= (   COORDINATE_REGION,
                        GEODETIC_REGION )
```

COORDINATE_REGION denotes an applicable region or extended region that is represented in terms of the coordinate system of the SRF.

GEODETIC_REGION denotes an applicable region or extended region that is represented in terms of the geodetic coordinate system of the Celestiodetic SRF for that ORM.

11.2.7 Selection data types

11.2.7.1 Overview

Selection data types are similar to enumerated data types but form a set of entries that may be extended through registration. Selection data types are defined to be distinct sub-data types of the numeric data type Integer, but with specific meanings attached to each value. Selection data types are otherwise processed in the same manner as enumerated data types. The integer codes are unique within each selection data type, but not between data types.

In each selection data type the valid values are 0 and greater. Negative code values are implementation dependent and non-conforming. In each selection data type, the value 0 (UNSPECIFIED) is reserved. Some API methods and functions allow 0 (UNSPECIFIED) as an input code value and/or an output code value. The valid use of 0 (UNSPECIFIED) is defined in the specification of the appropriate method or function.

11.2.7.2 CS_Code

The selection data type CS_Code specifies a CS by its code as defined in Clause 5 or by registration. Table 5.7 is a directory of CS specifications, each of which includes a code value and a corresponding label.

11.2.7.3 DSS_Code

The selection data type DSS_Code specifies a DSS by its code as defined in Table 9.2 and in Table J.20 or by registration. Each DSS specification includes a code value and a corresponding label.

11.2.7.4 ORM_Code

The selection data type `ORM_Code` specifies an ORM by its code as defined in Annex E and Annex J or by registration. Each ORM specification includes a code value and a corresponding label (see Clause 7).

11.2.7.5 ORMT_Code

The selection data type `ORMT_Code` specifies an ORM Template code defined in Clause 7 or by registration. Table 7.30 is a directory of ORMT specifications, each of which includes a code value and a corresponding label.

11.2.7.6 Profile_Code

The selection data type `Profile_Code` specifies a profile of the SRM by its code as defined in Clause 12 or by registration. Each profile specification includes a code value and a corresponding label.

11.2.7.7 RT_Code

The selection data type `RT_Code` specifies a reference transformation $\boldsymbol{H}_{\mathrm{R}\leftarrow\mathrm{S}}$. Each `RT_Code` is specified in Annex E in the entry for the ORM or by registration, specified by the `ORM_Code` value, with which it is associated. Each reference transformation specification associated with an ORM includes a code value and a corresponding label. An `RT_Code` is valid for an ORM only if it has been specified for that ORM. Some API methods also allow the `RT_Code` value `0 (UNSPECIFIED)` to be used.

API methods or functions that require the `RT_Code` data type shall also require its associated `ORM_Code`.

11.2.7.8 SRF_Code

The selection data type `SRF_Code` specifies an SRF by its code as defined in Table 8.31 or by registration. Each SRF specification includes a code value and a corresponding label (see Clause 8).

11.2.7.9 SRFS_Code

The selection data type `SRFS_Code` specifies an SRF set by its code as listed in Table 8.48 or by registration. Each SRF set specification includes a code value and a corresponding label (see Clause 8).

11.2.7.10 SRFS member types

11.2.7.10.1 Overview

The selection data types that specify the SRF set members associated with each of the SRF sets listed in Table 8.48.

11.2.7.10.2 Alabama_SPCS_Code

The selection data type `Alabama_SPCS_Code` specifies a member of the Alabama SPCS SRF set in Table 8.50 or by registration.

11.2.7.10.3 GTRS_Global_Coordinate_System_Code

The selection data type `GTRS_Global_Coordinate_System_Code` specifies a member of the GTRS Global Coordinate System SRF set in Table 8.52 and Table 8.53 or by registration.