11 Application program interface

11.1 Introduction

This International Standard specifies an API for the SRF operations in Clause 5 and Clause 10. The API specifies non-object data types (see 11.2), and object classes (see 11.3) used to perform the spatial operations. Two functions are provided to create certain object instances (see 11.4 and 11.5). Two query functions are also provided to indicate the extent of support of an API implementation for a profile of the SRM (see 11.6 and Clause 12). The API also specifies data storage structures for the representation of SRM concepts that are not used to perform spatial operations (see 11.9).

Class is the term used to categorize the general form of *object* instances. Each class definition specifies the *methods* (if any) that operate on the object. Methods are specified by giving their syntax (input and output parameters), semantics (how the inputs interact with the *state* of an instance of the class and produce any *outputs*), and error conditions. In particular, the state of an instance of the class is implicitly an input for each of its methods with the exception of the `Create` method. The `Create` method of an object depends only on its explicit inputs. The state of a class instance may change only as the result of applying a method of the class.

The active objects created as instances of a given class are reliably denoted by *object references*. Once created, objects exist and respond to method invocations until they are destroyed. The property of being created and existing until destruction is termed the *object life cycle*. Classes inherit methods from other classes through the *subclass*/*superclass* relationship. Method inheritance is transitive: a subclass also inherits the methods that have been inherited by its superclass.

Non-object data types do not have an object life cycle nor do they have operations other than those defined by a programming language that this API might be bound to.

EXAMPLE `Integer` is a non-object data type. Programming languages to which this API may be bound have definition mechanisms and operations for creating and then performing arithmetic operations on integers as variables and/or constants in the programming language.

The API specifies eight abstract classes (see 11.3.3 and 11.3.5):

a) `LifeCycleObject`,

b) `BaseSRF`,

c) `BaseSRF2D`,

d) `BaseSRF3D`,

e) `BaseSRFwithTangentPlaneSurface`,

f) `BaseSRFwithEllipsoidalHeight`,

g) `BaseSRFMapProjection`, and

h) `Orientation`.

These abstract classes are used as base classes from which subclasses including concrete classes inherit common sets of methods. `LifeCycleObject` includes the creation and destruction methods that all other classes inherit. The `LifeCycleObject` creation method specification may be overridden in a concrete subclass to provide subclass specific inputs and error conditions. The use of abstract classes in this International Standard is solely for the purpose of specifying common methods in only one place instead of repeating the same specification in each concrete class for which it applies. API implementations are not required to implement abstract classes.

The API specifies six classes whose methods are not exposed as part of the API:

a) three coordinate classes: `Coordinate2D`, `Coordinate3D`, and `SurfaceCoordinate`;

b) one direction class `Direction` and

c) two position classes: `Position2D`, and `Position3D`.

These classes are private classes that hide all aspects of the implementation of instances of these objects from the application.

The API specifies a set of concrete classes that correspond to specific SRFTs specified in Clause 8 (see 11.3.6 through 11.3.10). An instance of one of these concrete classes corresponds to a specific SRF.

The API specifies a set of concrete classes that correspond to different representations of orientations, as specified in 6.4 (see 11.3.11). An instance of one of these concrete classes corresponds to an instance of an orientation represented in a specific way (i.e., as a matrix or as a set of quaternions).

Instances of concrete SRF classes that correspond to the coded collection of SRFT instances specified in Table 8.30 are created by the function `CreateStandardSRF`. `CreateStandardSRF` takes the corresponding `SRF_Code` (see 11.4) as an input.

Instances of concrete SRF classes that correspond to the members of SRF Sets specified in Table 8.31 are also created by the function `CreateSRFSetMember`. `CreateSRFSetMember` takes an `SRFS_Code_Info` (see 11.5) as an input.

The class hierarchy is illustrated in Figure 11.1. Procedural rules for using `LifeCycleObject`s in applications and examples of use of the API are provided in 11.8.

11.2 Non-object data types

11.2.1 Overview

Basic non-object data types represent single pieces of information such as numbers, codes, and other individual data items. Structured data types represent data records of basic non-object data types.

11.2.2 Abbreviations

Table 11.1 lists the SRFTs and their abbreviations used in the formation of enumerant names and record element names of non-object types.

Table 11.1 — SRFT abbreviations

Abbreviation	SRFT
CC	Celestiocentric
CD	Celestiodetic
CM	Celestiomagnetic
EC	Equidistant Cylindrical
EI	Equatorial Inertial
HAEC	Heliospheric Aries Ecliptic
HEEC	Heliospheric Earth Ecliptic
HEEQ	Heliospheric Earth Equatorial

Abbreviation	SRFT
LCC	Lambert Conformal Conic
LCE_3D	Lococentric Euclidean 3D
LSA	Local Space Azimuthal
LSP	Local Space Polar
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
M	Mercator
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_Positive` and `Integer`. 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_Positive`	[1, 4 294 967 295]
`Integer`	[-2 147 483 647, 2 147 483 647]

`Long_Float` is a non-object data type defined for floating-point numbers. This data type corresponds to the double precision floating-point data type specified by IEC 60559. However, implementations on architectures that support other floating-point representations are allowed.

11.2.4 Logicals

The general-purpose logical data type is Boolean. All implementations that conform to this standard shall support this 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 an opaque non-object data type that allows an application to reliably access an instance of an object. Object_References may be compared for equality and tested to see if they are equal to the special value NULL_Object. If two Object_References are equal, they refer to the same object instance. If an Object_Reference is equal to the special value NULL_Object it does not reference any object instance. In all the method specifications in this clause, whenever an argument passed to or returned from a method is an object, it is an object reference that is passed.

11.2.6 Enumerated data types

11.2.6.1 Introduction

Enumerated data types are data types whose values are specified from an ordered list of names. The names are assigned numbers whose values indicate the position within the ordered list. It is these numbers that are actually manipulated by the implementation. Enumerated data types are a closed list the members of which do not change based on registration or deprecation. This clause specifies the enumerated data types within this International Standard.

11.2.6.2 Axis_Direction

This data type represents the values of the axis direction parameter(s) of the SRFTs LOCAL_SPACE_RECTANGULAR_3D and LOCAL_SPACE_RECTANGULAR_2D.

```
Axis_Direction ::= (  POSITIVE_PRIMARY_AXIS,
                      POSITIVE_SECONDARY_AXIS,
                      POSITIVE_TERTIARY_AXIS,
                      NEGATIVE_PRIMARY_AXIS,
                      NEGATIVE_SECONDARY_AXIS,
                      NEGATIVE_TERTIARY_AXIS )
```

11.2.6.3 Coordinate_Valid_Region

This data type represents coordinate location with respect to valid-regions (see 8.3.2.4).

```
Coordinate_Valid_Region ::= ( VALID,
                              EXTENDED_VALID,
                              DEFINED )
```

VALID denotes a coordinate that is contained in the valid-region and in the CS domain.
EXTENDED_VALID denotes a coordinate that is contained in the extended valid-region and in the CS domain but not in the valid-region.
DEFINED denotes a coordinate that is contained in the CS domain but not in the valid or the extended valid-regions.

11.2.6.4 Interval_Type

This data type is used to specify coordinate-component intervals in the SetValidRegion, SetExtendedValidRegion, GetValidRegion, and GetExtendedValidRegion methods of class BaseSRF3D and in the SetValidGeodeticRegion, SetExtendedValidGeodeticRegion, GetValidGeodeticRegion, and GetExtendedValidGeodeticRegion methods of class BaseSRFMapProjection.

```
Interval_Type::= (  OPEN_INTERVAL,      // The bounded open interval (a, b).
                    GE_LT_INTERVAL,     // The bounded interval [a, b).
                    GT_LE_INTERVAL,     // The bounded interval (a, b].
                    CLOSED_INTERVAL,    // The bounded interval [a, b].
                    GT_SEMI_INTERVAL,   // The unbounded interval (a, +infinity).
                    GE_SEMI_INTERVAL,   // The unbounded interval [a, +infinity).
                    LT_SEMI_INTERVAL,   // The unbounded interval (-infinity, b).
                    LE_SEMI_INTERVAL,   // The unbounded interval (-infinity, b].
                    UNBOUNDED           // All values (-infinity, +infinity)
                 )
```

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 )
```

11.2.7 Selection data types

11.2.7.1 Introduction

Selection data types are similar to enumerated data types but form a set of entries that may be extended. Selection data types are all defined to be as distinct sub-data types of the numeric of data type Integer, but with specific meanings attached to each value. The set of selections may be augmented by assigning meanings to additional values. Selection data types are otherwise processed in the same manner as enumerated data types. The integer codes are unique within each concept set, but not between sets. Although the RT_Code is used in combination with an ORM_Code, its code space follows the general rule and is independent of the ORM_Code.

In each code space the valid Integer values are 0 and greater. Negative code values are implementation dependent and non-conforming. In each code space, the Integer value 0 (UNSPECIFIED) is reserved. Some API methods and functions allow 0 (UNSPECIFIED) as an input Integer code value and/or an output Integer 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 Integer code 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 Integer code 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 `Integer` code 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 `Integer` code data type `ORMT_Code` specifies an ORM Template code defined in Clause 7 or by registration. Table 7.12 is a directory of ORMT specifications, each of which includes a code value and a corresponding label.

11.2.7.6 RT_Code

The `Integer` code data type `RT_Code` specifies a reference transformation H_{SR}. Each `RT_Code` is defined 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.

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

11.2.7.7 SRF_Code

The `Integer` code data type `SRF_Code` specifies an SRF by its code as defined in Table 8.30 or by registration. Each SRF specification includes a code value and a corresponding label (see Clause 8).

11.2.7.8 SRFS_Code

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

```
SRFS_Code ::= (      < 0 :   // implementation_dependent,
                       0 :   SRFS_UNSPECIFIED,
                       1 :   SRFS_ALABAMA_SPCS,
                       2 :   SRFS_GTRS_GLOBAL_COORDINATE_SYSTEM,
                       3 :   SRFS_JAPAN_RECTANGULAR_PLANE_CS,
                       4 :   SRFS_LAMBERT_NTF,
                       5 :   SRFS_UNIVERSAL_POLAR_STEREOGRAPHIC,
                       6 :   SRFS_UNIVERSAL_TRANSVERSE_MERCATOR,
                       7 :   SRFS_WISCONSIN_SPCS,
                      >7 :   // reserved for registration )
```

11.2.7.9 SRFS member types

11.2.7.9.1 Introduction

The `Integer` code types that specify the SRFS members associated with the SRFS defined in Table 8.48.

11.2.7.9.2 SRFSM_Alabama_SPCS_Code

The `Integer` code data type `SRFSM_Alabama_SPCS_Code` specifies a member of the Alabama SPCS SRFS in Table 8.50 or by registration.

11.2.7.9.3 SRFSM_GTRS_Global_Coordinate_System _Code

The `Integer` code data type `SRFSM_GTRS_Global_Coordinate_System_Code` specifies a member of the GTRS Global Coordinate System SRFS in Table 8.52 and Table 8.53 or by registration.

11.2.7.9.4 SRFSM_Japan_Rectangular_Plane_CS_Code

The `Integer` code data type `SRFSM_Japan_Rectangular_Plane_CS_Code` specifies a member of the Japan Rectangular Plane CS SRFS in Table 8.55 or by registration.

11.2.7.9.5 SRFSM_Lambert_NTF_Code

The `Integer` code data type `SRFSM_Lambert_NTF_Code` specifies a member of the Lambert NTF SRFS in Table 8.57 or by registration.

11.2.7.9.6 SRFSM_Universal_Polar_Stereographic_Code

The `Integer` code data type `SRFSM_Universal_Polar_Stereographic_Code` specifies a member of the Universal Polar Stereographic SRFS in Table 8.59 or by registration.

11.2.7.9.7 SRFSM_Universal_Transverse_Mercator_Code

The `Integer` code data type `SRFSM_Universal_Transverse_Mercator_Code` specifies a member of the Universal Transverse Mercator SRFS in Table 8.61 or by registration.

11.2.7.9.8 SRFSM_Wisconsin_SPCS _Code

The `Integer` code data type `SRFSM_Wisconsin_SPCS_Code` specifies a member of the Wisconsin SPCS SRFS Table 8.63 or by registration.

11.2.7.10 SRFT_Code

The `Integer` code data type `SRFT_Code` specifies an SRFT by its code as defined in Clause 8 or by registration. Table 8.3 is a directory of SRFT specifications. Each SRFT specification includes a code value and a corresponding label.

11.2.7.11 Status_Code

The `Status_Code` non-object selection data type specifies the status codes associated with methods on instances of classes specified in this International Standard. The meaning of values other than `SUCCESS` varies according to the class and method or function and is further defined in the “Error conditions” element of each method or function specification in Table 11.6 through Table 11.54 (see common error conditions in 11.3.2). This selection data type may be extended in a language binding specification.

```
Status_Code ::= (    < 0 :    // implementation_dependent,
                       0 :    UNSPECIFIED,    // reserved
                       1 :    SUCCESS,    // the operation was performed successfully
                       2 :    INVALID_SRF,
                       3 :    INVALID_SOURCE_SRF,
                       4 :    INVALID_SOURCE_COORDINATE,
                       5 :    INVALID_TARGET_COORDINATE,
                       6 :    INVALID_POINT1_COORDINATE,
                       7 :    INVALID_POINT2_COORDINATE,
                       8 :    OPERATION_UNSUPPORTED,
```

```
 9 :   INVALID_SOURCE_DIRECTION,
10 :   INVALID_TARGET_DIRECTION,
11 :   INVALID_CODE,
12 :   INVALID_INPUT,
13 :   CREATION_FAILURE,
14 :   DESTRUCTION_FAILURE,
15 :   FLOATING_OVERFLOW,
16 :   FLOATING_UNDERFLOW,
17 :   FLOATING_POINT_ERROR,
18 :   MEMORY_ALLOCATION_ERROR,
>18 :  // reserved for language binding specification )
```

11.2.7.12 STT_Code

The `Integer` code data type `STT_Code` specifies an STT by its code as defined in 7.3.3 or by registration. Each STT specification includes a code value and a corresponding label.

11.2.8 Array data types

11.2.8.1 Introduction

Array data types specify an ordered set whose elements may be of any single data type. Table 11.4 specifies the notation for Array data types.

Table 11.4 — Array data type notation

Data type	Notation
`One-dimensional array`	Data_Type_Name[length]
`Two-dimensional array`	Data_Type_Name[rows, columns]

The symbols “length”, "rows", and "columns" are positive integers. The length of a one-dimensional array is specified by "length". When the length is specified by another field of a record data type or by a function parameter, the field name or function parameter name that will be used to indicate that the size of the array is obtained from the value of that construct. The index of the first element in the array is either “0” or “1” depending on the language binding.

For two-dimensional arrays, "rows" and "columns" specify the number of rows and columns of the array respectively. The ordering of the set is row-major. The indices of the first element in the array are both either “0” or “1” depending on the language binding.

11.2.8.2 Coordinate2D_Array

This data type specifies an array of `Coordinate2D` objects.

```
Coordinate2D_Array ::= {
   length                        Integer_Positive;
   coordinate2D_array            Object_Reference[ length ];
}
```

11.2.8.3 Coordinate3D_Array

This data type specifies an array of `Coordinate3D` objects.

```
Coordinate3D_Array ::= {
   length                          Integer_Positive;
   coordinate3D_array              Object_Reference[ length ];
}
```

11.2.8.4 Coordinate_Valid_Region_Array

This data type specifies an array of `Coordinate_Valid_Region` variables.

```
Coordinate_Valid_Region_Array ::= {
   length                          Integer_Positive;
   valid_region_array              Coordinate_Valid_Region[ length ];
}
```

11.2.8.5 Direction_Array

This data type specifies an array of `Direction` objects.

```
Direction_Array ::= {
   length                          Integer_Positive;
   direction_array                 Object_Reference[ length ];
}
```

11.2.8.6 Vector_3D

This data type specifies an array of three `Long_Float` variables representing a vector in 3D Euclidean space.

```
Vector_3D ::= Long_Float[ 3 ]
```

11.2.8.7 Matrix_3x3

This data type specifies a two-dimensional square array of nine `Long_Float` variables representing a 3x3 matrix (see 10.4.6).

```
Matrix_3x3 ::= Long_Float[ 3, 3 ]
```

11.2.8.8 Matrix_4x4

This data type specifies a two-dimensional square array of 16 `Long_Float` variables representing a 4x4 matrix (see 10.4.6).

```
Matrix_4x4 ::= Long_Float[ 4, 4 ]
```

11.2.8.9 Matrix_2x2

This data type specifies a two-dimensional square array of 4 `Long_Float` variables representing a 2x2 matrix (see 10.4.6).

```
Matrix_2x2 ::= Long_Float[ 2, 2 ]
```

11.2.9 Structured data types

11.2.9.1 Introduction

Non-object data types created as records whose elements are basic non-object data types are called *structured non-object data types*. This International Standard specifies a set of structured non-object data types to collect the (non-ORM) parameters needed to specify an SRF by means of an SRF template, and to collect parameters needed to specify an ORM transformation.

The elements of structured data types that represent lengths shall be evaluated in the units of metre, and the elements that represent angles shall be evaluated in the units of radian.

The following notation is used for defining the variant record data structures for non-object types:

```
<Variant_Record_Data_Type> ::= ( <Selector_Name>   <Selection_Data_Type> )
{
  <Variable_Name>   <Variable_Data_Type>
  <Variable_Name>   <Variable_Data_Type>
  …
  [
    <Selection_Name> :   <Variable_Name>   <Variable_Data_Type>;
    <Selection_Name> :   <Variable_Name>   <Variable_Data_Type>;
    …
  ]
}
```

Where:

<Variant_Record_Data_Type>:	The variant record data type that is being defined.
<Selector_Name>:	The name of the selector
<Selector_Data_Type>:	The selection data type used to select the content of the variant record.
<Variable_Name>:	The name of a record element.
<Variable_Data_Type>:	The data type of a record element. Data type “<empty>” signifies the element is not present in the record.
<Selection_Name>:	A selection data type enumerant for which a record element applies.
{}:	The body of the variant record.
[]:	The variant part of the variant record.

11.2.9.2 SRFT parameters

11.2.9.2.1 EC_Parameters

This non-object data type specifies the parameters that correspond to SRFT EQUIDISTANT_CYLINDRICAL.

```
EC_Parameters ::= {
   origin_longitude              Long_Float;
   central_scale                 Long_Float;
   false_easting                 Long_Float;
   false_northing                Long_Float;
}
```

11.2.9.2.2 LCC_Parameters

This non-object data type specifies the parameters that correspond to SRFT LAMBERT_CONFORMAL_CONIC.

```
LCC_Parameters ::= {
   origin_longitude              Long_Float;
   origin_latitude               Long_Float;
   latitude1                     Long_Float;
   latitude2                     Long_Float;
   false_easting                 Long_Float;
   false_northing                Long_Float;
}
```

11.2.9.2.3 LSR_2D_Parameters

This non-object data type specifies the parameters that correspond to SRFT LOCAL_SPACE_RECTANGULAR_2D.

```
LSR_2D_Parameters ::= {
   forward_direction             Axis_Direction;
}
```

11.2.9.2.4 LSR_3D_Parameters

This non-object data type specifies the parameters that correspond to SRFT LOCAL_SPACE_RECTANGULAR_3D.

```
LSR_3D_Parameters ::= {
   forward_direction             Axis_Direction;
   up_direction                  Axis_Direction;
}
```

11.2.9.2.5 Local_Tangent_Parameters

This non-object data type specifies the parameters that correspond to SRFT LOCAL_TANGENT_SPACE_AZIMUTHAL_SPHERICAL, and SRFT LOCAL_TANGENT_SPACE_CYLINDRICAL.

```
Local_Tangent_Parameters ::={
   geodetic_longitude            Long_Float;
   geodetic_latitude             Long_Float;
   azimuth                       Long_Float;
   height_offset                 Long_Float;
}
```

11.2.9.2.6 LTSE_Parameters

This non-object data type specifies the parameters that correspond to SRFT LOCAL_TANGENT_SPACE_EUCLIDEAN.

```
LTSE_Parameters ::={
   geodetic_longitude            Long_Float;
   geodetic_latitude             Long_Float;
   azimuth                       Long_Float;
   x_false_origin                Long_Float;
   y_false_origin                Long_Float;
   height_offset                 Long_Float;
}
```

11.2.9.2.7 LCE_3D_Parameters

This non-object data type specifies the parameters that correspond to SRFT LOCOCENTRIC_EUCLIDEAN_3D.

```
LCE_3D_Parameters ::= {
   lococentre                    Vector_3D;
   primary_axis                  Vector_3D;
   secondary_axis                Vector_3D;
}
```

11.2.9.2.8 M_Parameters

This non-object data type specifies the parameters that correspond to SRFT MERCATOR.

```
M_Parameters ::= {
   origin_longitude              Long_Float;
   central_scale                 Long_Float;
   false_easting                 Long_Float;
   false_northing                Long_Float;
}
```

11.2.9.2.9 Oblique_Mercator_Parameters

This non-object data type specifies the parameters that correspond to SRFT OBLIQUE_MERCATOR_SPHERICAL.

```
Oblique_Mercator_Parameters ::= {
   longitude1                    Long_Float;
   latitude1                     Long_Float;
   longitude2                    Long_Float;
   latitude2                     Long_Float;
   central_scale                 Long_Float;
   false_easting                 Long_Float;
   false_northing                Long_Float;
}
```

11.2.9.2.10 PS_Parameters

This non-object data type specifies the parameters that correspond to SRFT POLAR_STEREOGRAPHIC.

```
PS_Parameters ::= {
   polar_aspect                  Polar_Aspect;
   origin_longitude              Long_Float;
   central_scale                 Long_Float;
   false_easting                 Long_Float;
   false_northing                Long_Float;
}
```

11.2.9.2.11 SRFS_Code_Info

This Variant_Record_Data_Type specifies an arbitrary SRFS_Code with its associated SRFS member code. The record element SRFSM_unspecified shall be set to zero (unspecified) when the selector value is SRFS_UNDEFINED.

```
SRFS_Code_Info ::= ( srfs_code    SRFS_Code )
{
   [
    SRFS_UNSPECIFIED:
      SRFSM_unspecified            Integer;
    SRFS_ALABAMA_SPCS:
      SRFSM_alabama_spcs           SRFSM_Alabama_SPCS_Code;
    SRFS_GTRS_GLOBAL_COORDINATE_SYSTEM:
      SRFSM_gtrs_global_coordinate_system
                                   SRFSM_GTRS_Global_Coordinate_System_Code;
    SRFS_JAPAN_RECTANGULAR_PLANE_CS:
      SRFSM_japan_rectangular_plane_cs
                                   SRFSM_Japan_Rectangular_Plane_CS_Code;
    SRFS_LAMBERT_NTF:
      SRFSM_lambert_ntf            SRFSM_Lambert_NTF_Code;
    SRFS_UNIVERSAL_POLAR_STEREOGRAPHIC:
      SRFSM_universal_polar_stereographic
                                   SRFSM_Universal_Polar_Stereographic_Code;
    SRFS_UNIVERSAL_TRANSVERSE_MERCATOR:
      SRFSM_universal_transverse_mercator
                                   SRFSM_Universal_Transverse_Mercator_Code;
    SRFS_WISCONSIN_SPCS:
      SRFSM_wisconsin_spcs         SRFSM_Wisconsin_SPCS_Code;
   ]
}
```

11.2.9.2.12 TM_Parameters

This non-object data type specifies the parameters that correspond to SRFT TRANSVERSE_MERCATOR.

```
TM_Parameters ::= {
   origin_longitude              Long_Float;
   origin_latitude               Long_Float;
   central_scale                 Long_Float;
   false_easting                 Long_Float;
   false_northing                Long_Float;
}
```

11.2.9.3 STT parameters

11.2.9.3.1 STT_Translate_3D_Parameters

This non-object data type represents the parameters of a 3D translation of the origin, as specified in Table 7.14.

```
STT_Translate_3D_Parameters ::= {
   delta_x                       Long_Float;
   delta_y                       Long_Float;
   delta_z                       Long_Float;
}
```

The values of `delta_x`, `delta_y`, and `delta_z` represent the origin displacement in metres.

11.2.9.3.2 STT_Translate_2D_Parameters

This non-object data type represents the parameters of a 3D translation of the origin as specified in Table 7.15.

```
STT_Translate_2D_Parameters ::= {
   delta_x                      Long_Float;
   delta_y                      Long_Float;

}
```

The values of `delta_x` and `delta_y` represent the origin displacement in metres.

11.2.9.3.3 STT_Rotate_Scale_Translate_3D_Parameters

This non-object data type represents the parameters of a general 3D transformation as specified in Table 7.16, Table 7.17, Table 7.23, and Table 7.24.

```
STT_Rotate_Scale_Translate_3D_Parameters ::= {
   delta_x                      Long_Float;
   delta_y                      Long_Float;
   delta_z                      Long_Float;
   omega_1                      Long_Float;
   omega_2                      Long_Float;
   omega_3                      Long_Float;
   delta_scale                  Long_Float;
}
```

The values of `delta_x`, `delta_y`, and `delta_z` represent the origin displacement in metres.

The values of `omega_1`, `omega_2`, and `omega_3` represent rotations in radians. The valid range for values of `omega_1`, `omega_2`, and `omega_3` is $(-2\pi, 2\pi)$.

The value of `delta_scale` represents a scale difference from unity. The valid range for `delta_scale` is greater than -1.

11.2.9.3.4 STT_Molodensky_Badekas_3D_Parameters

This non-object data type represents the parameters of a Molodensky-Badekas 3D transformation as specified in Table 7.18.

```
STT_Molodensky_Badekas_3D_Parameters ::= {
   delta_x                      Long_Float;
   delta_y                      Long_Float;
   delta_z                      Long_Float;
   omega_1                      Long_Float;
   omega_2                      Long_Float;
   omega_3                      Long_Float;
   delta_scale                  Long_Float;
   x_0                          Long_Float;
   y_0                          Long_Float;
   z_0                          Long_Float;
}
```

The values of `delta_x`, `delta_y`, and `delta_z` represent the origin displacement in metres.

The values of omega_1, omega_2, and omega_3 represent rotations in radians. The valid range for values of omega_1, omega_2, and omega_3 is (-2π, 2π).

The value of delta_scale represents a scale difference from unity. The valid range for delta_scale is greater than -1.

The values of x_0, y_0, and z_0 represent the initial point in metres.

11.2.9.3.5 STT_Similarity_3D_Parameters

This non-object data type represents the parameters of a 3D general similarity transformation, as specified in Table 7.19.

```
STT_Similarity_3D_Parameters ::= {
   delta_x                      Long_Float;
   delta_y                      Long_Float;
   delta_z                      Long_Float;
   matrix                       Matrix_3x3;
   scale                        Long_Float;
}
```

The values of delta_x, delta_y, and delta_z represent the origin displacement in metres.

The value of matrix represents a rotation matrix. Therefore, its inverse must equal its transpose, and its determinant must equal one.

The value of scale represents a scale factor. The valid range for scale is greater than zero.

11.2.9.3.6 STT_Similarity_2D_Parameters

This non-object data type represents the parameters of a 2D general similarity transformation, as specified in Table 7.20.

```
STT_Similarity_2D_Parameters ::= {
   delta_x                      Long_Float;
   delta_y                      Long_Float;
   matrix                       Matrix_2x2;
   scale                        Long_Float;
}
```

The values of delta_x and delta_y represent the origin displacement in metres.

The value of matrix represents a rotation matrix. Therefore, its inverse must equal its transpose, and its determinant must equal one.

The value of scale represents a scale factor. The valid range for scale is greater than 0.

11.2.9.3.7 STT_Homogeneous_3D_Parameters

This non-object data type represents the parameters of a 3D homogeneous transformation, as specified in Table 7.21.

```
STT_Homogeneous_3D_Parameters ::= {
   delta_x                        Long_Float;
   delta_y                        Long_Float;
   delta_z                        Long_Float;
   matrix                         Matrix_3x3;
}
```

The values of `delta_x`, `delta_y`, and `delta_z` represent the origin displacement in metres.

The value of `matrix` represents a scaled rotation matrix. Therefore, its determinant must be greater than zero, and its inverse must equal its transpose divided by the square of its determinant.

11.2.9.3.8 STT_Homogeneous_2D_Parameters

This non-object data type represents the parameters of a 2D homogeneous transformation, as specified in Table 7.22.

```
STT_Translation_2D_Parameters ::= {
   delta_x                        Long_Float;
   delta_y                        Long_Float;
   matrix                         Matrix_2x2;
}
```

The values of `delta_x` and `delta_y` represent the origin displacement in metres.

The value of `matrix` represents a scaled rotation matrix. Therefore, its determinant must be greater than zero, and its inverse must equal its transpose divided by the square of its determinant.

11.2.9.3.9 STT_Rotate_Translate_3D_Parameters

This non-object data type represents the parameters of a 3D translation of the origin and rotation about the axes as specified in Table 7.25 and Table 7.26.

```
STT_Rotate_Translate_3D_Parameters ::= {
   delta_x                        Long_Float;
   delta_y                        Long_Float;
   delta_z                        Long_Float;
   omega_1                        Long_Float;
   omega_2                        Long_Float;
   omega_3                        Long_Float;
}
```

The values of `delta_x`, `delta_y`, and `delta_z` represent the origin displacement in metres.

The values of `omega_1`, `omega_2`, and `omega_3` represent rotations in radians. The valid range for values of `omega_1`, `omega_2`, and `omega_3` is $(-2\pi, 2\pi)$.

11.2.9.3.10 STT_Z_Rotate_Translate_3D_Parameters

This non-object data type represents the parameters of a 3D translation of the origin and rotation about the z-axis as specified in Table 7.27.

```
STT_Z_Rotate_Translate_3D_Parameters ::= {
   delta_x                          Long_Float;
   delta_y                          Long_Float;
   delta_z                          Long_Float;
   omega_3                          Long_Float;
}
```

The values of `delta_x`, `delta_y`, and `delta_z` represent the origin displacement in metres.

The value of `omega_3` represents a rotation in radians about the z-axis. The valid range for values of `omega_3` is (-2π, 2π).

11.2.9.3.11 STT_Z_Rotate_3D_Parameters

This non-object data type represents the parameters of a 3D rotation about the z-axis as specified in Table 7.28.

```
STT_Z_Rotate_3D_Parameters ::= {
   omega_3                          Long_Float;
}
```

The value of `omega_3` represents a rotation in radians about the z-axis. The valid range for values of `omega_3` is (-2π, 2π).

11.2.9.3.12 STT_YZ_Rotate_3D_Parameters

This non-object data type represents the parameters of a 3D rotation about the y- and z-axes as specified in Table 7.29.

```
STT_YZ_Rotate_3D_Parameters ::= {
   omega_2                          Long_Float;
   omega_3                          Long_Float;
}
```

The values of `omega_2` and `omega_3` represent rotations in radians about the y- and z-axes. The valid range for values of `omega_2` and `omega_3` is (-2π, 2π).

11.2.9.3.13 STT_XZ_Rotate_3D_Parameters

This non-object data type represents the parameters of a 3D rotation about the x- and z-axes as specified in Table 7.30.

```
STT_XZ_Rotate_3D_Parameters ::= {
   omega_1                          Long_Float;
   omega_3                          Long_Float;
}
```

The values of `omega_1` and `omega_3` represent rotations in radians about the x- and z-axes. The valid range for values of `omega_1` and `omega_3` is (-2π, 2π).

11.2.9.3.14 ORM_Transformation_2D_Parameters

This variant record type represents a set of 2D ORM transformation parameters.

```
ORM_Transformation_2D_Parameters ::= { template_code STT_Code }
{
  [
  IDENTITY_2D:
    identity_2d_parameters <empty>;
  TRANSLATE_2D:
    translate_2d_parameters    STT_Translate_2D_Parameters;
  ROTATE_SCALE_TRANSLATE_2D:
    rotate_scale_translate_2d_parameters STT_Similarity_2D_Parameters;
  HOMOGENEOUS_MATRIX_3X3_2D:
    homogeneous_matrix_3x3_2d_parameters STT_Homogeneous_2D_Parameters;
  ]
}
```

11.2.9.3.15 ORM_Transformation_3D_Parameters

This variant record type represents a set of 3D ORM transformation parameters.

```
ORM_Transformation_3D_Parameters ::= { template_code STT_Code }
{
  [
  IDENTITY:
    identity_parameters <empty>;
  TRANSLATE:
    translate_parameters   STT_Translate_3D_Parameters;
  PV_7_PARAMETER:
    pv_7_parameters STT_Rotate_Scale_Translate_3D_Parameters;
  CF_7_PARAMETER:
    cf_7_parameters STT_Rotate_Scale_Translate_3D_Parameters;
  CF_7_PLUS_3_PARAMETER:
    cf_7_plus_3_parameters STT_Molodensky_Badekas_3D_Parameters;
  ROTATE_SCALE_TRANSLATE:
    rotate_scale_translate_parameters    STT_Similarity_3D_Parameters;
  HOMOGENEOUS_MATRIX_4X4:
    homogeneous_matrix_4x4_parameters    STT_Homogeneous_3D_Parameters;
  CF_XYZ_ROTATE_SCALE_TRANSLATE:
    cf_xyz_rotate_scale_translate_parameters
   STT_Rotate_Scale_Translate_3D_Parameters;
  PV_XYZ_ROTATE_TRANSLATE:
    pv_xyz_rotate_translate_parameters  STT_Rotate_Translate_3D_Parameters;
  CF_ZYX_ROTATE_TRANSLATE:
    cf_zyx_rotate_translate_parameters  STT_Rotate_Translate_3D_Parameters;
  PV_Z_ROTATE_TRANSLATE:
    pv_z_rotate_translate_parameters STT_Z_Rotate_Translate_3D_Parameters;
  CF_Z_ROTATE:
    cf_z_rotate_parameters STT_Z_Rotate_3D_Parameters;
  PV_YZ_ROTATE:
    pv_yz_rotate_parameters    STT_YZ_Rotate_3D_Parameters;
  CF_XZ_ROTATE:
    cf_xz_rotate_parameters    STT_XZ_Rotate_3D_Parameters;
  ]
}
```

11.2.9.4 Orientation representation parameters

11.2.9.4.1 Axis_Angle_Parameters

This non-object data type specifies the parameters that allow an Orientation object to be instantiated using an axis-angle representation. It consists of an axis, specified by a 3D vector, and a rotation angle about that axis. The vector is expressed in terms of its three components in the world SRF. The rotation angle is given in radians.

```
Axis_Angle_Parameters::= {
   axis                          Vector_3D;
   angle                         Long_Float;
}
```

11.2.9.4.2 Euler_Angles_ZXZ_Parameters

This non-object data type specifies the parameters that allow an Orientation object to be instantiated using an Euler angles ZXZ representation. It consists of three rotation angles in radians.

```
Euler_Angles_ZXZ_Parameters::= {
   spin                          Long_Float;
   nutation                      Long_Float;
   precession                    Long_Float;
}
```

11.2.9.4.3 Tait_Bryan_Angles_Parameters

This non-object data type specifies the parameters that allow an Orientation object to be instantiated using a Tait-Bryan angles representation. It consists of three rotation angles in radians.

```
Tait_Bryan_Angles_Parameters::= {
   roll                          Long_Float;
   pitch                         Long_Float;
   yaw                           Long_Float;
}
```

11.2.9.4.4 Matrix_3x3_Parameters

This non-object data type specifies the parameters that allow an Orientation object to be instantiated using a 3x3 rotation matrix representation. It consists of the nine matrix elements.

```
Matrix_3x3_Parameters::= {
   matrix                        Matrix_3x3;
}
```

11.2.9.4.5 Quaternion_Parameters

This non-object data type specifies the parameters that allow an Orientation object to be instantiated using a quaternion representation. It consists of a 4-tuple of numbers, the scalar part and the three vector parts. The parameter values must meet the constraint: $e_0^2 + e_1^2 + e_2^2 + e_3^2 = 1$.

```
Quaternion_Parameters::= {
   e0                              Long_Float;
   e1                              Long_Float;
   e2                              Long_Float;
   e3                              Long_Float;
}
```

11.3 Object classes

11.3.1 Introduction

SRF objects specify methods that implement the spatial operations specified in Clause 10. To aid in specification, most of the functionality of the API is defined using a class hierarchy with each abstract class providing the specification of those methods that are common to each of its subclasses. The remaining functionality is provided in concrete class and function specifications. The implementation of abstract classes is not required.

The functionality of the methods are specified in the class specification tables (see 11.3.2) that provide the method name, the semantics, inputs and outputs of the method, and the error conditions of the method. These methods manipulate internal data (object state) and any input parameters passed in. The success condition is a nominal behaviour of all methods and is not listed within the error conditions element. The success condition is associated with `Status_Code SUCCESS`.

EXAMPLE 1 In Table 11.13, the phrase "this SRF" refers to the internal state of an instance of a concrete class subclassed (directly or indirectly) from the abstract class specified in the table. In particular, the abstract method `GetORMCode` "Outputs the `ORM_Code` and the `RT_Code` of this SRF", and shows "Inputs: none".

Language bindings may add additional error conditions and related binding-specific mechanisms including the passing of inputs and outputs, and the presentation of method status. Language bindings shall specify these mechanisms, since this International Standard does not restrict such mechanisms. Under an error condition, output values are undefined. When several error conditions apply to a method invocation, the first error condition detected by an implementation shall be presented as the method status. The error conditions applicable to a method invocation are the common error conditions specified in 11.3.2 and the additional error conditions specified in the class specification table for the method and any language-binding specific error conditions applicable to the method.

A language binding mechanism for presentation of method status shall support the association of a unique error `Status_Code` (11.2.7.11)

EXAMPLE 2 If a language binding supports exception handling and if a language binding uses that mechanism to present method failure, then an exception object method that returns the corresponding `Status_Code` would satisfy this requirement.

11.3.2 Class specification format

Class data types are specified in tables in Table 11.5 through Table 11.54 with the following elements:

Table 11.5 — Class specification elements

Element	Definition
Class	The name of the object class.
Description	The corresponding SRM concept.

Element	Definition
Superclass(es)	The specification of inherited functionality listing the superclasses of the class in hierarchical order. Each superclass name is followed by a list of the methods it specifies. The method list excludes methods that are overridden.
Method or **Abstract method or** **Private method**	The name of the method.
Semantics	The specification of the method functionality.
Inputs	The specification(s) of the method input parameters, or "none". The state of the invoking object is implicitly an input and is not additionally listed in this element. The Create method of an object class is an exception. The Create method of an object class depends only on its explicit input parameters.
Outputs	The specification(s) of the method output parameters, or “none”.
Error conditions	The list of `Status_Code` values that correspond to error conditions. Each listed value specifies the condition for which it is applicable. Common error conditions (see below) are not listed in this element. The “success condition” is not listed in this element.

The method element of a concrete class is labelled “Method”. The method element of an abstract class is labelled “Abstract method” or “Private method”. A private method is used to document the functionality of other methods. A subclass inherits all the abstract methods of its superclass including methods that the superclass has inherited. In particular, an abstract method inherited by a concrete class shall be implemented for the concrete class. An implementation may implement private methods in concrete classes for internal use, but public access to a concrete implementation of a private method is not a requirement. The order in which the methods are listed in each class follows a life cycle pattern, i.e., creation methods, then data manipulation methods, and then spatial operations.

The Error conditions element of a method specification does not separately list the following *common error conditions*.

a) `INVALID_SRF` if the SRF object invoking the method was not successfully initialized by the API or is invalid. The condition does not apply to create methods.

b) `FLOATING_OVERFLOW` if a floating-point overflow error occurred in the performance of the method.

c) `FLOATING_UNDERFLOW` if a floating-point underflow error occurred in the performance of the method.

d) `FLOATING_POINT_ERROR` on input, if a `Long_Float` input is positive or negative infinity, or a not-a-number (NAN) value. On output, if a floating-point error (other than an overflow or underflow error) occurred in the performance of the method.

e) `MEMORY_ALLOCATION_ERROR` if a memory allocation error occurred in the performance of the method.

11.3.3 LifeCycleObject

The `LifeCycleObject` class is the most basic abstract class from which all other classes inherit. `LifeCycleObject` is an abstract class. All other abstract classes are defined in 11.3.5.

Table 11.6 — LifeCycleObject

Element	Specification
Class	`LifeCycleObject`
Description	This is the most basic abstract class from which all other classes inherit. An object derived from a concrete subclass of `LifeCycleObject` is invalid until the `Create` method is successfully invoked. A valid object is invalid after the `Destroy` method is invoked.
Superclass(es)	None, this is a top level object.
Abstract method	`Create`
Semantics	This method creates an instance of a concrete class. An implementation may perform memory allocation and/or object initialization as part of this method.
Inputs	Specific inputs are specified in concrete classes that are directly or indirectly subclassed from this class data type.
Outputs	`object_reference:` `Object_Reference`
Error conditions	`CREATION_FAILURE` if the object instance could not be created.
Abstract method	`Destroy`
Semantics	This method destroys an instance of a concrete class. An implementation may perform memory deallocation and/or object finalization as part of this method.
Inputs	`object_reference:` `Object_Reference`
Outputs	none
Error conditions	`DESTRUCTION_FAILURE` if the object was not successfully deallocated.

11.3.4 Private objects

11.3.4.1 Introduction

Private object classes are concrete classes that represent objects whose methods and attributes are not exposed to the API user. Instances of these objects may be created by methods on other objects specified in the API. Object references for instances of private objects may be passed to and returned from methods. This allows an implementation of the API to store data that can be maintained for these private object classes in whatever form is convenient for the implementation.

11.3.4.2 Coordinate3D

The `Coordinate3D` class represents a coordinate of CS data type 3D. It specifies no additional public methods.

Table 11.7 — Coordinate3D

Element	Specification
Class	Coordinate3D
Description	A coordinate of CS data type 3D (see 5.3).
Superclass(es)	LifeCycleObject: Create, Destroy

11.3.4.3 Coordinate2D

The Coordinate2D class represents a coordinate of CS data type 2D. It specifies no additional public methods.

Table 11.8 — Coordinate2D

Element	Specification
Class	Coordinate2D
Description	A coordinate of CS data type 2D (see 5.3).
Superclass(es)	LifeCycleObject: Create, Destroy

11.3.4.4 SurfaceCoordinate

The SurfaceCoordinate class represents a coordinate of CS data type surface. It specifies no additional public methods.

Table 11.9 — SurfaceCoordinate

Element	Specification
Class	SurfaceCoordinate
Description	A coordinate of CS data type surface (see 5.5.2).
Superclass(es)	LifeCycleObject: Create, Destroy

11.3.4.5 Direction

The Direction class encapsulates an SRF representation of a spatial direction including the reference position coordinate and a vector in the local tangent frame of the SRF at the reference position. It specifies no additional public methods.

Table 11.10 — Direction

Element	Specification
Class	Direction
Description	A direction (see 10.5.2).
Superclass(es)	LifeCycleObject: Create, Destroy

11.3.4.6 Position3D

The Position3D class represents a point in a 3D position-space. It specifies no additional public methods. It is the input or output of some private methods of the BaseSRF3D class.

Table 11.11 — Position3D

Element	Specification
Class	Position3D
Description	A point in a 3D position-space.
Superclass(es)	LifeCycleObject: Create, Destroy

11.3.4.7 Position2D

The Position2D class represents a point in a 2D position-space. It specifies no additional public methods. It is the input or output of some private methods of the BaseSRF2D class.

Table 11.12 — Position2D

Element	Specification
Class	Position2D
Description	A point in a 2D position-space.
Superclass(es)	LifeCycleObject: Create, Destroy

11.3.5 Abstract classes

11.3.5.1 BaseSRF

This is the base class for all SRF classes. BaseSRF provides common methods to return coded values that may be associated with an SRF class instance.

Table 11.13 — BaseSRF

Element	Specification
Class	BaseSRF
Description	An abstract class specifying the common methods of all SRF classes.
Superclass(es)	LifeCycleObject: Create, Destroy
Abstract method	GetORMCodes
Semantics	Outputs the ORM_Code and the RT_Code of this SRF.
Inputs	none
Outputs	orm_code: ORM_Code rt_code: RT_Code
Error conditions	No additional error conditions. (See 11.3.2.)
Abstract method	GetSRFCodes
Semantics	1) Outputs the SRFT_Code of this SRF class instance.

Element	Specification
	2) If created by the CreateStandardSRF function, outputs a valid SRF_Code (otherwise output 0). (See 11.4.) 3) If created by the CreateSRFSetMember function, outputs a valid SRFS_Code_Info (otherwise outputs the SRFS_Code_Info with SRFS_Code selector value set to SRFS_UNSPECIFIED). (See 11.5.)
Inputs	none
Outputs	srft_code: SRFT_Code srf_code: SRF_Code srfs_code_info: SRFS_Code_Info
Error conditions	No additional error conditions. (See 11.3.2.)
Abstract method	GetCSCode
Semantics	Outputs the CS_Code code of this SRF.
Inputs	none
Outputs	cs_code: CS_Code
Error conditions	No additional error conditions. (See 11.3.2.)

11.3.5.2 BaseSRF2D

This is the base class for all 2D SRF classes. BaseSRF2D is a subclass of BaseSRF. This abstract class adds the following methods:

CreateCoordinate2D,
GetCoordinate2DValues,
ChangeCoordinate2DSRF,
ChangeCoordinate2DSRFObject,
ChangeCoordinate2DArraySRF,
ChangeCoordinate2DArraySRFObject, and
EuclideanDistance.

These methods, respectively, create a Coordinate2D, query component values of an existing Coordinate2D, change Coordinate2D representation from some other 2D SRF instance, change an array of Coordinate2D representation from some other 2D SRF instance, or compute the Euclidean distance between coordinates.

The ChangeCoordinate2DSRF method is for the case of source and target SRF instances based on object-fixed ORMs for the same spatial object using the methodology of 10.4.2 and the ORM reference transformations specified by the RT_Code input to the concrete SRF Create method. The method ChangeCoordinate2DSRFObject is for the general case of any pair of instances of concrete SRFs subclassed from BaseSRF2D where the ORM transformation $\boldsymbol{H}_{\mathbf{ST}}$ is explicitly specified with an input ORM_Transformation_2D_Parameters data structure. The ChangeCoordinate2DArraySRF and the ChangeCoordinate2DArraySRFObject methods are similar to the above operating on arrays of coordinates.

BaseSRF2D also adds the private methods Generating2D and InverseGenerating2D that are used to specify the functionality of the ChangeCoordinate2DSRF, and ChangeCoordinate2DSRFObject methods.

Table 11.14 — BaseSRF2D

Element	Specification
Class	BaseSRF2D
Description	An abstract class representing the common elements of concrete SRF classes that have a coordinate system of CS data type 2D.
Superclass(es)	LifeCycleObject: Create, Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode
Abstract method	CreateCoordinate2D
Semantics	This method accepts the two ordered coordinate-components and for a specific SRF concrete class instance creates a 2D coordinate instance initialized with the values passed in. Coordinate-components that represent lengths shall be evaluated in metres. Coordinate-components that represent angles shall be evaluated in radians.
Inputs	first_coordinate_component: Long_Float second_coordinate_component: Long_Float
Outputs	new_coordinate: Coordinate2D
Error conditions	INVALID_INPUT if the coordinate values are not in the accuracy domain of this SRF.
Abstract method	GetCoordinate2Dvalues
Semantics	This method retrieves the two ordered coordinate-components of a 2D coordinate instance previously initialized through the API. Coordinate-components that represent lengths shall be evaluated in metres. Coordinate-components that represent angles shall be evaluated in radians.
Inputs	coordinate: Coordinate2D
Outputs	first_coordinate_component: Long_Float second_coordinate_component: Long_Float
Error conditions	INVALID_SOURCE_COORD if the coordinate is not associated with this SRF, or not initialized through the API.
Abstract method	ChangeCoordinate2DSRF
Semantics	This method changes the SRF representation of the spatial position specified by the input Coordinate2D source_coordinate in the source SRF source_srf to a Coordinate2D target_coordinate in this SRF, the target SRF, in accordance with 10.4.2 using the implicit ORM transformation $\boldsymbol{H}_{\mathbf{ST}}$ given in Equation (10.3). The required functionality is equivalent to: 1) apply the Generating2D method of srf_source to the input source_coordinate to obtain a source Position2D as an output, 2) apply the implicit $\boldsymbol{H}_{\mathbf{ST}}$ to the source Position2D to obtain a target Position2D, and 3) apply the InverseGenerating2D method of this SRF to the target Position2D to obtain the target_coordinate output.
Inputs	source_srf: BaseSRF2D source_coordinate: Coordinate2D
Outputs	target_coordinate: Coordinate2D

Element	Specification
Error conditions	1) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 2) INVALID_SOURCE_COORDINATE if source_coordinate is not (1) in the accuracy domain of the source_srf, or (2) associated with the source_srf. 3) INVALID_TARGET_COORDINATE if the target_coordinate is not in the accuracy domain of this SRF.
Abstract method	ChangeCoordinate2DSRFObject
Semantics	This method changes the SRF representation of the spatial position specified by the input Coordinate2D source_coordinate in the source SRF source_srf to a Coordinate2D target_coordinate in this SRF, the target SRF, using an explicit ORM transformation $\boldsymbol{H}_{ST}$ as specified by the h_st input in accordance with subclause 10.4.2. The required functionality is equivalent to: 1) apply the Generating2D method of srf_source to the input source_coordinate to obtain a source Position2D as an output, 2) apply the explicit $\boldsymbol{H}_{ST}$ to the source Position2D to obtain a target Position2D, and 3) apply the InverseGenerating2D method of this SRF to the target Position2D to obtain the target_coordinate output.
Inputs	source_srf: BaseSRF2D source_coordinate: Coordinate2D h_st: ORM_Transformation_2D_Parameters
Outputs	target_coordinate: Coordinate2D
Error conditions	1) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 2) INVALID_SOURCE_COORDINATE if source_coordinate is not (1) in the accuracy domain of the source_srf, or (2) associated with the source_srf. 3) INVALID_INPUT if h_st parameter values are not in range. 4) INVALID_TARGET_COORDINATE if the target_coordinate is not in the accuracy domain of this SRF.
Abstract method	ChangeCoordinate2DArraySRF
Semantics	This method performs the same operation defined for the ChangeCoordinate2DSRF method on each Coordinate2D object in the source_coordinate_array. The processing is in array indexing order. Upon an error condition, the processing is halted and the output index is set to the array index of the offending coordinate. When successful, the output index is set to the size of the array plus one.
Inputs	source_srf: BaseSRF2D source_coordinate_array: Coordinate2D_Array
Outputs	target_coordinate_array: Coordinate2D_Array index: Integer_Positive

Element	Specification
Error conditions	1) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 2) INVALID_SOURCE_COORDINATE if the coordinate element index in the source_coordinate_array is not (1) in the accuracy domain of the source_srf, or (2) associated with the source_srf. 3) INVALID_INPUT if the source_coordinate_array and the target_coordinate_array are not the same size. 4) INVALID_TARGET_COORDINATE if the coordinate element index in the target_coordinate_array is not (1) in the accuracy domain of this SRF, or (2) associated with this SRF.
Abstract method	ChangeCoordinate2DArraySRFObject
Semantics	This method performs the same operation defined for the ChangeCoordinate2DSRFObject method on each Coordinate2D object in the input source_coordinate_array. The processing is in array indexing order. Upon an error condition, the processing is halted and the output index is set to the array index of the offending coordinate. When successful, the output index is set to the size of the array plus one.
Inputs	source_srf: BaseSRF2D source_coordinate_array: Coordinate2D_Array h_st: ORM_Transformation_2D_Parameters
Outputs	target_coordinate_array: Coordinate2D_Array index: Integer_Positive
Error conditions	1) INVALID_SOURCE_COORDINATE if the coordinate element index in the source_coordinate_array is not (1) in the accuracy domain of the source_srf, or (2) associated with the source_srf. 2) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 3) INVALID_INPUT if the source_coordinate_array and the target_coordinate_array are not the same size, or if the h_st is not valid. 4) INVALID_TARGET_COORDINATE if the coordinate element index in the target_coordinate_array is not (1) in the accuracy domain of this SRF, or (2) associated with this SRF.
Abstract method	EuclideanDistance
Semantics	Outputs the Euclidean distance in metres between the spatial points represented by Coordinate2D point1_coordinate and point2_coordinate.
Inputs	point1_coordinate: Coordinate2D point2_coordinate: Coordinate2D
Outputs	distance: Long_Float
Error conditions	1) INVALID_POINT1_COORDINATE if point1_coordinate is not in the coordinate system domain of this SRF. 2) INVALID_POINT2_COORDINATE if point2_coordinate is not in the coordinate system domain of this SRF.
Private method	Generating2D
Semantics	This method implements the coordinate system generating function of this SRF changing the input Coordinate2D in coordinate-space to the output Position2D in position-space as specified in 5.9.

Element	Specification
Inputs	coordinate: Coordinate2D
Outputs	position: Position2D
Error conditions	INVALID_SOURCE_COORDINATE if coordinate is not in the coordinate system domain of this SRF.
Private method	InverseGenerating2D
Semantics	This method implements the coordinate system inverse generating function of this SRF changing the input Position2D in position-space to the output Coordinate2D in coordinate-space as specified in 5.9.
Inputs	position: Position2D
Outputs	coordinate: Coordinate2D
Error conditions	INVALID_SOURCE_COORDINATE if coordinate is not in the coordinate system domain of this SRF.

NOTE The method ChangeCoordinate2DSRF semantics describes the required functionality in terms of the private methods Generating2D, and InverseGenerating2D. This computational scheme is presented only for the purpose of specifying the required functionality, and does not take into account error checking and various computational short cuts and efficiencies that an actual implementation design would consider. (Some of these efficiencies are presented in Clause 10). This remark is also applicable to the ChangeCoordinate2DSRFObject method.

11.3.5.3 BaseSRF3D

This is the base class for 3D SRF classes. BaseSRF3D is a subclass of BaseSRF. This abstract class adds the methods SetValidRegion and SetExtendedValidRegion to define coordinate valid and extended valid-regions to an SRF as in 8.3.2.4, and GetValidRegion and GetExtendedValidRegion to retrieve this information.

BaseSRF3D adds the following methods:

CreateCoordinate3D,
GetCoordinate3DValues,
ChangeCoordinate3DSRF,
ChangeCoordinate3DSRFObject,
ChangeCoordinate3DArraySRF,
ChangeCoordinate3DArraySRFObject, and
EuclideanDistance.

These methods, respectively, create a Coordinate3D, query component values of an existing Coordinate3D, change Coordinate3D representation from some other 3D SRF instance, change an array of Coordinate3D representation from some other 3D SRF instance, or compute the Euclidean distance between coordinates.

BaseSRF3D also adds the following methods:

CreateDirection,
GetDirectionValues,
ChangeDirectionSRF,
ChangeDirectionSRFObject,
ChangeDirectionArraySRF, and
ChangeDirectionArraySRFObject.

These methods, respectively, create a Direction, query component values of an existing Direction, change Direction representation from some other 3D SRF instance, or change an array of Direction representation from some other 3D SRF instance.

The ChangeCoordinate3DSRF and ChangeDirectionSRF methods are for source and target SRF instances based on object-fixed ORMs for the same spatial object using the methodology of 10.3.1 and the ORM reference transformations specified by the RT_Code input to the concrete SRF Create method. The methods ChangeCoordinate3DSRFObject and ChangeDirectionSRFObject are for the general case of any pair of instances of concrete SRF classes subclassed from BaseSRF3D where the ORM transformation $\boldsymbol{H}$ is explicitly specified with an input ORM_Transformation_3D_Parameters data structure.

This class adds the method CreateLococentricEuclidean3DSRF to create a LococentricEuclidean3D SRF with a specified lococentre and axis directions. BaseSRF3D also adds the private methods Generating3D and InverseGenerating3D that are used to specify the functionality of the ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeDirectionSRFObject, and ChangeDirectionSRF methods.

The class BaseSRF3D also adds the following methods:

```
ComputeSRFOrientation,
GetLocalTangentFrameSRFParameters,
TransformOrientation,
TransformOrientationCommonOrigin,
TransformVector,
TransformVectorCommonOrigin,
TransformVectorInBodyFrame,
TransformVectorInBodyFrameCommonOrigin
```

These methods compute orientations and vectors in the local tangent frame associated with the specified reference location in the target SRF, given a corresponding orientation or vector in the local tangent frame associated with the specified reference location in the source SRF. The methods TransformOrientationCommonOrigin, TransformVectorCommonOrigin, and TransformVectorInBodyFrameCommonOrigin use the same reference location for both the source and target SRFs. The methods TransformVectorInBodyFrame and TransformVectorInBodyFrameCommonOrigin allow the source vector to be specified in terms of a body frame rather than a local tangent frame, by specifying the orientation of the body frame with respect to a local tangent frame.

Table 11.15 — BaseSRF3D

Element	Specification
Class	BaseSRF3D
Description	An abstract class representing the common elements of concrete SRF classes that have a coordinate system of CS data type 3D.
Superclass(es)	LifeCycleObject: Create, Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode
Abstract method	CreateCoordinate3D
Semantics	This method creates a Coordinate3D for this SRF from three ordered coordinate-component values. Coordinate-components that represent lengths shall be evaluated in metres. Coordinate-components that represent angles shall be evaluated in radians.

Element	Specification
Inputs	first_coordinate_component: Long_Float second_coordinate_component: Long_Float third_coordinate_component: Long_Float
Outputs	new_coordinate: Coordinate3D
Error conditions	INVALID_INPUT if the coordinate values are not in the accuracy domain of this SRF.
Abstract method	SetValidRegion
Semantics	This method creates a numeric interval for valid values of a coordinate-component. (See 8.3.2.4.) Given a coordinate-component, the last invocation of this method or the SetExtendedValidRegion method determines the valid interval of the coordinate-component values. If an extended value interval has previously been set for a coordinate-component by the SetExtendedValidRegion method, the extended interval limits shall be adjusted as necessary to contain the valid interval. The input value of upper_bound_bound is ignored if type is a semi-interval GT_SEMI_INTERVAL, or GE_SEMI_INTERVAL, or UNBOUNDED. The input value of lower_bound is ignored if type is a semi-interval LT_SEMI_INTERVAL, or LE_SEMI_INTERVAL, or UNBOUNDED.
Inputs	component_identifier: Integer interval_type: Interval_Type lower_bound: Long_Float upper_bound_bound: Long_Float
Outputs	none
Error conditions	INVALID_INPUT if a) the value of component_identifier is not 1, 2, or 3, or b) the coordinate-component is not angular and the value of lower_bound is not strictly less than the value of upper_bound_bound and type is not a semi-interval or unbounded type, or c) the coordinate-component is angular and the value of type is a semi-interval type, or d) the coordinate-component is angular, the value of type is a bounded interval and the value of lower_bound or upper_bound_bound does not satisfy the corresponding CS domain constraints or the values are equal.
Abstract method	SetExtendedValidRegion
Semantics	This method creates numeric intervals for both valid values and extended valid values of a coordinate-component. (See 8.3.2.4.) Given a coordinate-component, the last invocation of this method or the SetValidRegion method determines the valid and extended valid intervals of the coordinate-component values. The upper_bound and extended_upper_bound values are ignored if type is GT_SEMI_INTERVAL, or GE_SEMI_INTERVAL, or UNBOUNDED. The lower_bound and extended_lower_bound values are ignored if type is LT_SEMI_INTERVAL, or LE_SEMI_INTERVAL, or UNBOUNDED.

Element	Specification
Inputs	component_identifier: Integer interval_type: Interval_Type extended_lower_bound: Long_Float lower_bound: Long_Float upper_bound: Long_Float extended_upper_bound: Long_Float
Outputs	none
Error conditions	INVALID_INPUT if a) the value of component_identifier is not 1, 2, or 3, or b) the coordinate-component is not angular and the value of lower_bound is not strictly less than the value of upper_bound and type is not a semi-interval or unbounded type, or c) the coordinate-component is angular and the value of type is a semi-interval type, or d) the coordinate-component is angular, the value of type is a bounded interval and the value of lower_bound or upper_bound does not satisfy the corresponding CS domain constraints or the values are equal, or e) the interval determined by the values extended_lower_bound and extended_upper_bound does not contain the valid region.
Abstract method	GetValidRegion
Semantics	This method returns the numeric interval for valid values of a coordinate-component that have been set in the last SetValidRegion or SetExtendedValidRegion method invocation for the same coordinate-component. If an interval has not been set for the coordinate-component, the output type shall be the Interval_Type UNBOUNDED. If the output type is GT_SEMI_INTERVAL, or GE_SEMI_INTERVAL, the output value of upper_bound shall be 0.0. If the output type is LT_SEMI_INTERVAL, or LE_SEMI_INTERVAL, the output value of lower_bound shall be 0.0. If the output type is UNBOUNDED, the output values of lower_bound and upper_bound shall be 0.0.
Inputs	component_identifier: Integer
Outputs	interval_type: Interval_Type lower_bound: Long_Float upper_bound: Long_Float
Error conditions	INVALID_INPUT if the value of component_identifier is not 1, 2, or 3.

Element	Specification
Abstract method	GetExtendedValidRegion
Semantics	This method returns the numeric intervals for valid and extended values of a coordinate-component that have been set in the last SetValidRegion or SetExtendedValidRegion method invocation for the same coordinate-component. If an interval has not been set for the coordinate-component, the output type shall be the Interval_Type UNBOUNDED. If SetExtendedValidRegion has not been invoked for the coordinate-component, then the output extended_lower_bound shall equal the output lower_bound and the output extended_upper_bound shall equal the output upper_bound. If the output type is GT_SEMI_INTERVAL, or GE_SEMI_INTERVAL, the output values of upper_bound and extended_upper_bound shall be 0.0. If the output type is LT_SEMI_INTERVAL, or LE_SEMI_INTERVAL, the output values of lower_bound and extended_lower_bound shall be 0.0. If the output type is UNBOUNDED, the output values of lower_bound, extended_lower_bound, upper_bound and extended_upper_bound shall be 0.0.
Inputs	component_identifier: Integer
Outputs	interval_type: Interval_Type extended_lower_bound: Long_Float lower_bound: Long_Float upper_bound: Long_Float extended_upper_bound: Long_Float
Error conditions	INVALID_INPUT if the value of component_identifier is not 1, 2, or 3.
Abstract method	CreateDirection
Semantics	This method accepts a reference Coordinate3D and the three direction components of a direction vector in the local tangent frame of this SRF at reference Coordinate3D. The output is a Direction instance initialized with the input values. The direction vector shall be a unit vector.
Inputs	reference_coordinate: Coordinate3D first_direction_component: Long_Float second_direction_component: Long_Float third_direction_component: Long_Float
Output	new_direction: Direction
Error conditions	1) INVALID_SOURCE_COORD if reference_coordinate is not in the accuracy domain of this SRF. 2) INVALID_SOURCE_DIRECTION If the direction components are all zero.
Abstract method	GetCoordinate3Dvalues
Semantics	This method retrieves the three ordered coordinate-components of a 3D coordinate instance previously initialized through the API. Coordinate-components that represent lengths shall be evaluated in metres. Coordinate-components that represent angles shall be evaluated in radians.
Inputs	coordinate: Coordinate3D

Element	Specification
Outputs	first_coordinate_component: Long_Float second_coordinate_component: Long_Float third_coordinate_component: Long_Float
Error conditions	INVALID_SOURCE_COORD if the coordinate was not a 3D coordinate for this SRF, or not initialized through the API.
Abstract method	GetDirectionValues
Semantics	With an input Direction instance previously initialized through the API, this method retrieves the reference coordinate and the three components of a direction vector in the local tangent frame of this SRF at the reference coordinate.
Inputs	direction: Direction
Outputs	reference_coordinate: Coordinate3D first_direction_component: Long_Float second_direction_component: Long_Float third_direction_component: Long_Float
Error conditions	INVALID_SOURCE_DIRECTION if the reference coordinate was not a 3D coordinate for this SRF, or not initialized through the API.
Abstract method	ChangeCoordinate3DSRF
Semantics	This method changes the SRF representation of the spatial position specified by the input Coordinate3D source_coordinate in the source SRF source_srf to a Coordinate3D target_coordinate in this SRF, the target SRF, in accordance with subclause 10.4.2 using the implicit ORM transformation $\boldsymbol{H}_{ST}$ given in Equation (10.3). The required functionality is equivalent to: 1) apply the Generating3D method of srf_source to the input source_coordinate to obtain a source Position3D as an output, 2) apply the implicit $\boldsymbol{H}_{ST}$ to the source Position3D to obtain a target Position3D, and 3) apply the InverseGenerating3D method of this SRF to the target Position3D to obtain the target_coordinate and region outputs. When successful, the region output is the enumerated value DEFINED, unless a valid-region or extended valid-region has been defined in terms of coordinate intervals for this SRF using the SetValidRegion or SetExtendedValidRegion methods. In that case, the appropriate enumerated Coordinate_Valid_Region value is returned.
Inputs	source_srf: BaseSRF3D source_coordinate: Coordinate3D
Outputs	target_coordinate: Coordinate3D region: Coordinate_Valid_Region

Element	Specification
Error conditions	1) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 2) INVALID_SOURCE_COORDINATE if source_coordinate is not in the accuracy domain of the SRF specified by source_srf. 3) OPERATION_UNSUPPORTED If this SRF or the source_srf was created with reference transformation RT_Code value 0 (UNSPECIFIED), or if source_srf is an SRF for a different spatial object. 4) INVALID_TARGET_COORDINATE if the spatial position is not in the accuracy domain of this SRF.
Abstract method	ChangeCoordinate3DSRFObject
Semantics	This method changes the SRF representation of the spatial position specified by the input Coordinate3D source_coordinate in the source SRF source_srf to a Coordinate3D target_coordinate in this SRF, the target SRF, using an explicit ORM transformation $\boldsymbol{H}_{ST}$ as specified by the h_st input in accordance with subclause 10.4.1. The required functionality is equivalent to: 1) apply the Generating3D method of srf_source to the input source_coordinate to obtain a source Position3D as an output, 2) apply the explicit $\boldsymbol{H}_{ST}$ to the source Position3D to obtain a target Position3D, and 3) apply the InverseGenerating3D method of this SRF to the target Position3D to obtain the target_coordinate and region outputs. When successful, the region output is the enumerated value DEFINED, unless a valid-region of extended valid-region has been defined in terms of coordinate intervals for this SRF. In that case, the appropriate enumerated Coordinate_Valid_Region value is returned.
Inputs	source_srf: BaseSRF3D source_coordinate: Coordinate3D h_st: ORM_Transformation_3D_Parameters
Outputs	target_coordinate: Coordinate3D region: Coordinate_Valid_Region
Error conditions	1) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 2) INVALID_SOURCE_COORDINATE if source_coordinate is not in the accuracy domain of the SRF specified by source_srf. 3) INVALID_INPUT if h_st parameter values are not in range. 4) INVALID_TARGET_COORDINATE if the spatial position is not in the accuracy domain of this SRF.
Abstract method	ChangeCoordinateArray3DSRF
Semantics	This method performs the same operation defined for the ChangeCoordinate3DSRF method on each Coordinate3D object in the input source_coordinate_array. The processing is in array indexing order. Upon an error condition, the processing is halted and the output index is set to the array index of the offending coordinate. When successful, the output index is set to the size of the array plus one.
Inputs	source_srf: BaseSRF3D source_coordinate_array: Coordinate3D_Array

Element	Specification
Outputs	target_coordinate_array: Coordinate3D_Array index: Integer_Positive region_array: Coordinate_Valid_Region_Array
Error conditions	1) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 2) INVALID_SOURCE_COORDINATE if the coordinate element index in the source_coordinate_array is not (1) in the accuracy domain of the SRF source_srf, or (2) associated with the source_srf SRF. 3) INVALID_INPUT if the source_coordinate_array, the target_coordinate_array, and the region_array are not the same size. 4) OPERATION_UNSUPPORTED If this SRF or the source_srf was created with reference transformation RT_Code value 0 (UNSPECIFIED). 5) INVALID_TARGET_COORDINATE if the coordinate element index in the target_coordinate_array is not (1) in the accuracy domain of this SRF, or (2) associated with this SRF.
Abstract method	ChangeCoordinate3DArraySRFObject
Semantics	This method performs the same operation defined for the ChangeCoordinate3DSRFObject method on each Coordinate3D object in the input source_coordinate_array. The processing is in array indexing order. Upon an error condition, the processing is halted and the output index is set to the array index of the offending coordinate. When successful, the output index is set to the size of the array plus one.
Inputs	source_srf: BaseSRF3D source_coordinate_array: Coordinate3D_Array h_st: ORM_Transformation_3D_Parameters
Outputs	target_coordinate_array: Coordinate3D_Array index: Integer_Positive region_array: Coordinate_Valid_Region_Array
Error conditions	1) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 2) INVALID_SOURCE_COORDINATE if the coordinate element index in the source_coordinate_array is not (1) in the accuracy domain of the SRF source_srf, or (2) associated with the source_srf SRF. 3) INVALID_INPUT if the source_coordinate_array, the target_coordinate_array, and the region_array are not the same size, or if the h_st parameter values are not in range. 4) INVALID_TARGET_COORDINATE if the coordinate element index in the target_coordinate_array is not (1) in the accuracy domain of this SRF, or (2) associated with this SRF.

Element	Specification
Abstract method	ChangeDirectionSRF
Semantics	This method changes the input source_direction, a spatial direction represented in the source SRF source_srf, to its corresponding representation, target_direction, in this SRF, the target SRF. The output Direction is computed in accordance with 10.5.3 using the implicit ORM transformation $\boldsymbol{H}_{ST}$ given in Equation (10.3) based on the RT_Codes of the source and target SRFs. The reference_coordinate-component of the output target_direction is computed from the reference_coordinate-component of the input source_direction. The reference_coordinate-component computation is functionally equivalent to ChangeCoordinate3DSRF. When successful, the ref_coord_region output is the enumerated value DEFINED, unless a valid-region of extended valid-region has been defined in terms of coordinate intervals for this SRF. In that case, the appropriate enumerated value is returned to correspond to the reference_coordinate-component of the target_direction.
Inputs	source_srf: BaseSRF3D source_direction: Direction
Outputs	target_direction: Direction ref_coord_region: Coordinate_Valid_Region
Error conditions	1) INVALID_SOURCE_DIRECTION If the reference coordinate of the Direction is invalid or if the direction components are all zero. 2) INVALID_SOURCE_SRF, if source_srf was not successfully initialized through the API or is invalid. 3) OPERATION_UNSUPPORTED, if this SRF or the source_srf was created with reference transformation RT_Code value 0 (UNSPECIFIED), or if source_srf is an SRF for a different spatial object. 4) INVALID_SOURCE_COORDINATE, if the reference location of the direction is not in the accuracy domain of this source_srf. 5) INVALID_TARGET_COORDINATE if the reference location of the direction is not in the accuracy domain of this SRF.

Element	Specification
Abstract method	ChangeDirectionSRFObject
Semantics	This method changes the input source_direction, a spatial direction represented in the source SRF source_srf, to its corresponding representation, target_direction, in this SRF, the target SRF. The output Direction is computed in accordance with 10.5.3 using the rotation matrix of the explicit ORM transformation $\boldsymbol{H}_{ST}$ corresponding to the h_st input. The reference_coordinate-component of the output target_direction is computed from the reference_coordinate-component of the input source_direction. The reference_coordinate-component computation is functionally equivalent to ChangeCoordinate3DSRF. When successful, the ref_coord_region output is the enumerated value DEFINED, unless a valid-region of extended valid-region has been defined in terms of coordinate intervals for this SRF. In that case, the appropriate enumerated value is returned to correspond to the reference_coordinate-component of the target_direction.
Inputs	source_srf: BaseSRF3D source_direction: Direction h_st: ORM_Transformation_3D_Parameters
Outputs	target_direction: Direction ref_coord_region: Coordinate_Valid_Region
Error conditions	1) INVALID_SOURCE_DIRECTION If the reference coordinate of the source_direction is invalid or if the direction components are all zero. 2) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 3) INVALID_INPUT if h_st parameter values are not in range. 4) INVALID_SOURCE_COORDINATE if the reference location of the direction is not in the accuracy domain of this source_srf. 5) INVALID_TARGET_COORDINATE if the reference location of the direction is not in the accuracy domain of this SRF.
Abstract method	ChangeDirectionArraySRF
Semantics	This method performs the same operation defined for the ChangeDirectionSRF method on each Direction object in the input source_direction_array. The processing is in array indexing order. Upon an error condition, the processing is halted and the output index is set to the array index of the offending coordinate. When successful, the output index is set to the size of the array plus one.
Inputs	source_srf: BaseSRF3D source_direction_array: Direction_Array
Outputs	target_direction_array: Direction_Array index: Integer_Positive ref_coord_region_array: Coordinate_Valid_Region_Array

Element	Specification
Error conditions	1) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 2) INVALID_SOURCE_DIRECTION if the direction element index in the source_direction_array (1) has its reference coordinate outside the accuracy domain of the source_srf SRF or (2) has its direction components all zeros, or (3) is not associated with the source_srf SRF. 3) INVALID_INPUT if the source_direction_array, the target_direction_array, and the ref_coord_region_array are not the same size. 4) OPERATION_UNSUPPORTED If this SRF or the source_srf was created with reference transformation RT_Code value 0 (UNSPECIFIED). 5) INVALID_TARGET_DIRECTION if the direction element index in the target_direction_array (1) has its reference coordinate outside the accuracy domain of this SRF, or (2) is not associated with this SRF.
Abstract method	ChangeDirectionArraySRFObject
Semantics	This method performs the same operation defined for the ChangeDirectionSRFObject method on each Direction object in the input source_direction_array. The processing is in array indexing order. Upon an error condition, the processing is halted and the output index is set to the array index of the offending coordinate. When successful, the output index is set to the size of the array plus one.
Inputs	source_srf: BaseSRF3D source_direction_array: Direction_Array h_st: ORM_Transformation_3D_Parameters
Outputs	target_direction_array: Direction_Array index: Integer_Positive ref_coord_region_array: Coordinate_Valid_Region_Array
Error conditions	1) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 2) INVALID_SOURCE_DIRECTION if the direction element index in the source_direction_array (1) has its reference coordinate outside the accuracy domain of the source_srf SRF or (2) has its direction components all zeros, or (3) is not associated with the source_srf SRF. 3) INVALID_INPUT if the source_direction_array, the target_direction_array, and the ref_coord_region_array are not the same size, or if the h_st parameter values are not in range. 4) INVALID_TARGET_DIRECTION if the direction element index in the target_direction_array (1) has its reference coordinate outside the accuracy domain of this SRF, or (2) is not associated with this SRF.
Abstract method	EuclideanDistance
Semantics	Outputs the Euclidean distance (in metres) between the spatial points represented by Coordinate3D point1_coordinate and point2_coordinate.
Inputs	point1_coordinate: Coordinate3D point2_coordinate: Coordinate3D
Outputs	distance: Long_Float

Element	Specification
Error conditions	1) INVALID_POINT1_COORDINATE if point1_coordinate is not in the coordinate system domain of this SRF. 2) INVALID_POINT2_COORDINATE if point2_coordinate is not in the coordinate system domain of this SRF.
Abstract method	CreateLococentricEuclidean3DSRF
Semantics	Creates a LococentricEuclidean3D SRF with origin at the input Coordinate3D and orthogonal axes determined by the input Direction parameters. The created SRF has the same ORM as this SRF.
Inputs	lococentre: Coordinate3D primary_axis: Direction secondary_axis: Direction
Outputs	lococentricEuclidean3D_srf: LococentricEuclidean3D
Error conditions	1) INVALID_SOURCE_COORDINATE if lococentre is not a valid coordinate in this SRF. 2) INVALID_SOURCE_DIRECTION if for either primary_axis and/or secondary_axis, the reference coordinate was not a 3D coordinate for this SRF, or not initialized through the API. 3) INVALID_INPUT if primary_axis and secondary_axis are not orthogonal directions.
Private method	Generating3D
Semantics	This method implements the coordinate system generating function of this SRF changing the input Coordinate3D in coordinate-space to the output Position3D in position-space as specified in 5.9.
Inputs	coordinate: Coordinate3D
Outputs	position: Position3D
Error conditions	INVALID_SOURCE_COORDINATE if coordinate is not in the accuracy domain of this SRF.
Private method	InverseGenerating3D
Semantics	This method implements the coordinate system inverse generating function of this SRF changing the input Position3D in position-space to the output Coordinate3D in coordinate-space as specified in 5.9. When successful, the region output is the enumerated value DEFINED, unless a valid-region of extended valid-region has been defined in terms of coordinate intervals for this SRF. In that case, the appropriate enumerated value is returned.
Inputs	position: Position3D
Outputs	coordinate: Coordinate3D region: Coordinate_Valid_Region
Error conditions	INVALID_SOURCE_COORDINATE if coordinate is not in the accuracy domain of this SRF.
Abstract method	ComputeSRFOrientation
Semantics	This method sets the values of the Orientation object representing the orientation of the source SRF (source_srf) at the source reference location (source_ref_location) with respect to this SRF at the target reference location (target_ref_location).

Element	Specification
Inputs	source_srf: BaseSRF3D source_ref_location: Coordinate3D target_ref_location: Coordinate3D
Outputs	out_orientation: Orientation
Error conditions	1) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 2) INVALID_SOURCE_COORDINATE if source_ref_location is not (1) in the accuracy domain of the source_srf, or (2) associated with the source_srf. 3) INVALID_TARGET_COORDINATE if the target_ref_location is not in the accuracy domain of this SRF.
Abstract method	GetLocalTangentFrameSRFParameters
Semantics	This method computes the parameters corresponding to the local tangent frame at the input reference location.
Inputs	reference_location: Coordinate3D
Outputs	ltf_parameters: LCE_3D_Parameters
Error conditions	1) INVALID_SOURCE_COORDINATE if reference_location is not (1) in the accuracy domain of this SRF, or (2) associated with this SRF.
Abstract method	TransformOrientation
Semantics	Given an orientation with respect to a local tangent frame (LTF_S) associated with a reference location in the source SRF, this method computes the orientation with respect to the local tangent frame (LTF_T) associated with the specified reference location in the target SRF. The output orientation is computed by composing the orientation of LTF_S with respect to LTF_T with the input source orientation. The SRF invoking this method is the target SRF.
Inputs	source_srf: BaseSRF3D source_ref_location: Coordinate3D source_orientation: Orientation target_ref_location: Coordinate3D
Outputs	target_orientation: Orientation
Error conditions	1) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 2) INVALID_SOURCE_COORDINATE if source_ref_location is not (1) in the accuracy domain of the source_srf, or (2) associated with the source_srf. 3) INVALID_TARGET_COORDINATE if the target_ref_location is not in the accuracy domain of this SRF.
Abstract method	TransformOrientationCommonOrigin
Semantics	Given an orientation with respect to a local tangent frame (LTF_S) associated with a reference location in the source SRF, this method computes the orientation with respect to the local tangent frame (LTF_T) associated with the specified reference location in the target SRF. LTF_S and LTF_T have a common origin. The output orientation is computed by composing the orientation of LTF_S with respect to LTF_T with the input source orientation. The SRF invoking this method is the target SRF.

Element	Specification
Inputs	source_srf: BaseSRF3D source_ref_location: Coordinate3D source_orientation: Orientation
Outputs	target_ref_location: Coordinate3D target_orientation: Orientation valid_region_category: Coordinate_Valid_Region
Error conditions	1) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 2) INVALID_SOURCE_COORDINATE if source_ref_location is not (1) in the accuracy domain of the source_srf, or (2) associated with the source_srf.
Abstract method	TransformVector
Semantics	Given a vector in the local tangent frame (LTF_S) associated with a reference location in the source SRF, this method computes the vector in the local tangent frame (LTF_T) associated with the specified reference location in the target SRF. The output vector is computed by applying the orientation of LTF_S with respect to LTF_T to the source vector. The SRF invoking this method is the target SRF.
Inputs	source_srf: BaseSRF3D source_ref_location: Coordinate3D source_vector: Vector_3D target_ref_location: Coordinate3D
Outputs	target_vector: Vector_3D
Error conditions	1) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 2) INVALID_SOURCE_COORDINATE if source_ref_location is not (1) in the accuracy domain of the source_srf, or (2) associated with the source_srf. 3) INVALID_TARGET_COORDINATE if the target_ref_location is not in the accuracy domain of this SRF.
Abstract method	TransformVectorCommonOrigin
Semantics	Given a vector in the local tangent frame (LTF_S) associated with a reference location in the source SRF, this method computes the vector in the local tangent frame (LTF_T) associated with the specified reference location in the target SRF. LTF_S and LTF_T have a common origin. The output vector is computed by applying the orientation of LTF_S with respect to LTF_T to the source vector. The SRF invoking this method is the target SRF.
Inputs	source_srf: BaseSRF3D source_ref_location: Coordinate3D source_vector: Vector_3D
Outputs	target_ref_location: Coordinate3D target_vector: Vector_3D valid_region_category: Coordinate_Valid_Region

Element	Specification
Error conditions	1) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 2) INVALID_SOURCE_COORDINATE if source_ref_location is not (1) in the accuracy domain of the source_srf, or (2) associated with the source_srf.
Abstract method	TransformVectorInBodyFrame
Semantics	Given a vector in a body frame (or in general any linear reference frame) whose orientation with respect to the local tangent frame (LTF_S) that is associated with a reference location in the source SRF, this method computes the vector in the local tangent frame (LTF_T) associated with the specified reference location in the target SRF. The output vector is computed by applying the composed orientation, from the orientation of LTF_S with respect to LTF_T with the source orientation, to the source vector. This method is equivalent to applying the orientation result from the transformOrientation method to the source vector. The SRF invoking this method is the target SRF.
Inputs	source_srf: BaseSRF3D source_ref_location: Coordinate3D source_orientation: Orientation source_vector: Vector_3D target_ref_location: Coordinate3D
Outputs	target_vector: Vector_3D
Error conditions	1) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 2) INVALID_SOURCE_COORDINATE if source_ref_location is not (1) in the accuracy domain of the source_srf, or (2) associated with the source_srf. 3) INVALID_TARGET_COORDINATE if the target_ref_location is not in the accuracy domain of this SRF.
Abstract method	TransformVectorInBodyFrameCommonOrigin
Semantics	Given a vector in a body frame (or in general any linear reference frame) whose orientation with respect to the local tangent frame (LTF_S) that is associated with a reference location in the source SRF, this method computes the vector in the local tangent frame (LTF_T) associated with the specified reference location in the target SRF. LTF_S and LTF_T have a common origin. The output vector is computed by applying the composed orientation, from the orientation of LTF_S with respect to LTF_T with the source orientation, to the source vector. This method is equivalent to applying the orientation result from the transformOrientation method to the source vector. The SRF invoking this method is the target SRF.
Inputs	source_srf: BaseSRF3D source_ref_location: Coordinate3D source_orientation: Orientation source_vector: Vector_3D
Outputs	target_ref_location: Coordinate3D target_vector: Vector_3D valid_region_category: Coordinate_Valid_Region

Element	Specification
Error conditions	1) INVALID_SOURCE_SRF if source_srf was not successfully initialized through the API or is invalid. 2) INVALID_SOURCE_COORDINATE if source_ref_location is not (1) in the accuracy domain of the source_srf, or (2) associated with the source_srf.

NOTE The method ChangeCoordinate3DSRF semantics describes the required functionality in terms of the private methods Generating3D, and InverseGenerating3D. This computational scheme is presented only for the purpose of specifying the required functionality, and does not take into account error checking and various computational short cuts and efficiencies that an actual implementation design would consider. (Some of these efficiencies are presented in Clause 10). This remark is also applicable to the ChangeCoordinate3DSRFObject, ChangeDirectionSRF and ChangeDirectionSRFObject methods.

11.3.5.4 BaseSRFwithTangentPlaneSurface

This is the base class for the following SRF classes:

LocalTangentSpaceEuclidian,
LocalTangentSpaceAzimuthalSpherical, and
LocalTangentSpaceCylindrical.

BaseSRFwithTangentPlaneSurface is derived from the BaseSRF3D class. It adds methods for the surface coordinate system that is induced on the vertical coordinate-component zero surface. That surface is a tangent plane to the oblate ellipsoid RD of the ORM of the SRF. The added methods for SurfaceCoordinate operations are CreateSurfaceCoordinate, GetSurfaceCoordinateValues, to create and query values of a SurfaceCoordinate, and AssociateSurfaceCoordinate and PromoteSurfaceCoordinate to relate a SurfaceCoordinate coordinate to Coordinate3D coordinate. The EuclideanDistance method of BaseSRF3D operates on a pair of Coordinate3D coordinates. This class adds a version of EuclideanDistance to operate on a pair of SurfaceCoordinate coordinates.

Table 11.16 — BaseSRFwithTangentPlaneSurface

Element	Specification
Class	BaseSRFwithTangentPlaneSurface
Description	An abstract class representing the common elements of BaseSRF3D concrete subclasses for which the vertical coordinate-component surface for zero is a tangent plane to the oblate ellipsoid RD of the ORM of the SRF. A surface coordinate system is induced on the third coordinate surface for zero.
Superclass(es)	LifeCycleObject: Create, Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D

Element	Specification
Abstract method	CreateSurfaceCoordinate
Semantics	Creates a surface coordinate on the tangent plane surface. Coordinate-components that represent lengths shall be evaluated in metres. Coordinate-components that represent angles shall be evaluated in radians.
Inputs	first_coordinate_component: Long_Float second_coordinate_component: Long_Float
Outputs	new_coordinate: SurfaceCoordinate
Error conditions	INVALID_INPUT if the input values are not in the domain of the induced surface CS generating function as specified in 5.9.
Abstract method	GetSurfaceCoordinateValues
Semantics	Retrieves the component values of a surface coordinate on the tangent plane surface. Coordinate-components that represent lengths shall be evaluated in metres. Coordinate-components that represent angles shall be evaluated in radians.
Inputs	coordinate: SurfaceCoordinate
Outputs	first_coordinate_component: Long_Float second_coordinate_component: Long_Float
Error conditions	INVALID_SOURCE_COORDINATE if coordinate is not a surface coordinate in this SRF, or not initialized through the API.
Abstract method	AssociateSurfaceCoordinate
Semantics	Creates the SurfaceCoordinate associated with a Coordinate3D by setting the third coordinate-component to zero and then converting that coordinate to its surface coordinate system representation (Truncate to surface, see 10.4.3).
Inputs	coordinate: Coordinate3D
Outputs	on_surface_coordinate: SurfaceCoordinate
Error conditions	INVALID_SOURCE_COORDINATE if coordinate_3D is not a valid 3D coordinate in this SRF.
Abstract method	PromoteSurfaceCoordinate
Semantics	Creates a Coordinate3D representing the same location as specified by surface_coordinate (Promote surface coordinate to 3D coordinate, see 10.4.3).
Inputs	surface_coordinate: SurfaceCoordinate
Outputs	coordinate: Coordinate3D
Error conditions	INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate in this SRF.
Abstract method	EuclideanDistance
Semantics	Outputs the Euclidean distance (in metres) between the spatial points represented by SurfaceCoordinate point1_coordinate and point2_coordinate.
Inputs	point1_coordinate: SurfaceCoordinate point2_coordinate: SurfaceCoordinate
Outputs	distance: Long_Float

Element	Specification
Error conditions	1) INVALID_POINT1_COORDINATE if point1_coordinate is not in the coordinate system domain of this SRF. 2) INVALID_POINT2_COORDINATE if point2_coordinate is not in the coordinate system domain of this SRF.

11.3.5.5 BaseSRFwithEllipsoidalHeight

This is the base class for the Celestiodetic SRF class, the Planetodetic SRF class, and the BaseSRFMapProjection class. BaseSRFwithEllipsoidalHeight is derived from the BaseSRF3D class. It adds methods for the surface coordinate system that is induced on the zero ellipsoidal height surface. The added methods for SurfaceCoordinate operations are CreateSurfaceCoordinate, GetSurfaceCoordinateValues, to create and query values of a SurfaceCoordinate, and AssociateSurfaceCoordinate and PromoteSurfaceCoordinate to relate a SurfaceCoordinate coordinate to Coordinate3D coordinate. The EuclideanDistance method of BaseSRF3D operates on Coordinate3D coordinates.

This class adds a version of EuclideanDistance to operate on SurfaceCoordinate coordinates. It also adds the GeodesicDistance method that returns the geodesic distance between a pair of SurfaceCoordinate coordinates; the GeodesicDistanceWithAzimuths method that returns the geodesic distance as well as the forward azimuths at the initial and destination points; and the GeodesicDestination method that returns the destination SurfaceCoordinate coordinate and forward azimuth, given an initial SurfaceCoordinate coordinate, forward azimuth, and distance (see 10.7).

The class also adds the method CreateLocalTangentSpaceEuclideanSRF to create a LocalTangentSpaceEuclidean SRF on a point on the zero ellipsoidal height surface. And it adds the VerticalOffset method for vertical offset models (see 9.3).

Table 11.17 — BaseSRFwithEllipsoidalHeight

Element	Specification
Class	BaseSRFwithEllipsoidalHeight
Description	An abstract class representing the common elements of BaseSRF3D concrete subclasses that have an ORM with an oblate ellipsoid RD and a coordinate system of CS data type 3D with ellipsoidal height (based on the RD) as the vertical coordinate-component. A surface coordinate system is induced on the vertical coordinate surface for zero.
Superclass(es)	LifeCycleObject: Create, Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D

Element	Specification
Abstract method	CreateSurfaceCoordinate
Semantics	Creates a surface coordinate on the ellipsoid RD surface. Coordinate-components that represent lengths shall be evaluated in metres. Coordinate-components that represent angles shall be evaluated in radians.
Inputs	first_coordinate_component: Long_Float second_coordinate_component: Long_Float
Outputs	new_coordinate: SurfaceCoordinate
Error conditions	INVALID_INPUT if the input values are not in the domain of the induced surface CS generating function as specified in 5.9.
Abstract method	GetSurfaceCoordinateValues
Semantics	Retrieves the component values of a surface coordinate on the ORM surface. Coordinate-components that represent lengths shall be evaluated in metres. Coordinate-components that represent angles shall be evaluated in radians.
Inputs	coordinate: SurfaceCoordinate
Outputs	first_coordinate_component: Long_Float second_coordinate_component: Long_Float
Error conditions	INVALID_SOURCE_COORDINATE if coordinate is not a surface coordinate in this SRF, or not initialized through the API.
Abstract method	AssociateSurfaceCoordinate
Semantics	Creates the SurfaceCoordinate associated with a Coordinate3D by setting the third coordinate-component to zero and then converting that coordinate to its Surface CS representation (Truncate to surface, see 10.4.3).
Inputs	coordinate: Coordinate3D
Outputs	on_surface_coordinate: SurfaceCoordinate
Error conditions	INVALID_SOURCE_COORDINATE if coordinate_3D is not a valid 3D coordinate in this SRF.
Abstract method	PromoteSurfaceCoordinate
Semantics	Creates a Coordinate3D representing the same location as specified by surface_coordinate (Promote surface coordinate to coordinate 3D, see 10.4.3).
Inputs	surface_coordinate: SurfaceCoordinate
Outputs	coordinate: Coordinate3D
Error conditions	INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate in this SRF.
Abstract method	CreateLocalTangentSpaceEuclideanSRF
Semantics	Creates a LocalTangentSpaceEuclidian SRF with natural origin at the input SurfaceCoordinate. The created SRF has the same ORM as this SRF. The input surface_coordinate determines the tangent point geodetic parameters. The created SRF origin is determined from the remaining input parameters. Input parameters that represent lengths shall be evaluated in the units of metre. The azimuth parameter shall be evaluated in the units of radian.

Element	Specification
Inputs	surface_coordinate: SurfaceCoordinate azimuth: Long_Float false_x_origin: Long_Float false_y_origin: Long_Float offset_height: Long_Float
Outputs	localtangentEuclidian_srf: LocalTangentSpaceEuclidean
Error conditions	INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate in this SRF.
Abstract method	VerticalOffset
Semantics	Outputs the vertical offset (see 9.3) at the input surface coordinate between the ellipsoid RD of this SRF and the DSS model specified by a DSS_code. If the DSS_code value is 0 (UNSPECIFIED), the output separation value shall be 0.
Inputs	DSS_code: DSS_Code surface_coordinate: SurfaceCoordinate
Outputs	separation: Long_Float
Error conditions	1) INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate for this SRF. 2) INVALID_CODE if the DSS_code is not a valid DSS_code or 0. 3) UNSUPPORTED_OPERATION if the DSS is not a VOS for this SRF, or if the DSS does not have a supported DSS model, or if the vertical offset is undefined at the input surface location.
Abstract method	GeodesicDistance
Semantics	Outputs the geodesic distance in metres between a pair of positions on the surface of the ellipsoid RD (*i.e.*, solves the geodesic indirect problem).
Inputs	point1_coordinate: SurfaceCoordinate point2_coordinate: SurfaceCoordinate
Outputs	distance: Long_Float
Error conditions	1) INVALID_POINT1_COORDINATE if point1_coordinate is not a valid surface coordinate for this SRF. 2) INVALID_POINT2_COORDINATE if point2_coordinate is not a valid surface coordinate for this SRF.
Abstract method	GeodesicDistanceWithAzimuths
Semantics	Outputs the geodesic distance in metres between a pair of positions on the surface of the ellipsoid RD, as well as the forward azimuths at both positions (*i.e.*, solves the geodesic indirect problem).
Inputs	point1_coordinate: SurfaceCoordinate point2_coordinate: SurfaceCoordinate
Outputs	distance: Long_Float point1_forward_azimuth: Long_Float point2_forward_azimuth: Long_Float
Error conditions	1) INVALID_POINT1_COORDINATE if point1_coordinate is not a valid surface coordinate for this SRF. 2) INVALID_POINT2_COORDINATE if point2_coordinate is not a valid surface coordinate for this SRF.

Element	Specification
Abstract method	GeodesicDestination
Semantics	Outputs the destination position on the surface of the ellipsoid RD, as well as the forward azimuth at the destination position, given the starting position, the forward azimuth at the starting position, and the distance to the destination *(i.e.*, solves the geodesic direct problem).
Inputs	point1_coordinate: SurfaceCoordinate point1_forward_azimuth: Long_Float distance: Long_Float
Outputs	point2_coordinate: SurfaceCoordinate point2_forward_azimuth: Long_Float
Error conditions	1) INVALID_POINT1_COORDINATE if point1_coordinate is not a valid non-polar surface coordinate for this SRF. 2) INVALID_INPUT if the distance value is non-positive.
Abstract method	EuclideanDistance
Semantics	Outputs the Euclidean distance (in metres) between the spatial points represented by the SurfaceCoordinates point1_coordinate and point2_coordinate.
Inputs	point1_coordinate: SurfaceCoordinate point2_coordinate: SurfaceCoordinate
Outputs	distance: Long_Float
Error conditions	1) INVALID_POINT1_COORDINATE if point1_coordinate is not in the coordinate system domain of this SRF. 2) INVALID_POINT2_COORDINATE if point2_coordinate is not in the coordinate system domain of this SRF.

11.3.5.6 BaseSRFMapProjection

This is the base class for concrete map projection SRF classes. BaseSRFMapProjection is derived from the BaseSRFwithEllipsoidalHeight class. This abstract class adds the following methods:

SetValidGeodeticRegion,
SetExtendedValidGeodeticRegion,
GetValidGeodeticRegion, and
GetExtendedValidGeodeticRegion

These methods define and retrieve coordinate valid and extended valid-regions with surface geodetic coordinate-component intervals in addition to map projection coordinate-component intervals that can be specified using the SetValidRegion and SetExtendedValidRegion methods of BaseSRF3D.

The BaseSRFMapProjection class also adds the following map projection specific methods:

ConvergenceOfTheMeridian,
PointDistortion, and
MapAzimuth.

Table 11.18 — BaseSRFMapProjection

Element	Specification
Class	`BaseSRFMapProjection`
Description	An abstract class representing the common elements of `BaseSRFwithEllipsoidalHeight` concrete subclasses have a map projection coordinate system.

Superclass(es)	`LifeCycleObject`: `Create`, `Destroy`
	`BaseSRF`: `GetORMCodes`, `GetSRFCodes`, `GetCSCode`
	`BaseSRF3D`: `CreateCoordinate3D`, `SetValidRegion`, `SetExtendedValidRegion`, `GetValidRegion`, `GetExtendedValidRegion`, `CreateDirection`, `GetCoordinate3Dvalues`, `GetDirectionValues`, `ChangeCoordinate3DSRF`, `ChangeCoordinate3DSRFObject`, `ChangeCoordinate3DArraySRF`, `ChangeCoordinate3DArraySRFObject`, `ChangeDirectionSRF`, `ChangeDirectionSRFObject`, `ChangeDirectionArraySRF`, `ChangeDirectionArraySRFObject`,`EuclideanDistance`, `CreateLococentricEuclidean3DSRF`, `Generating3D`, `InverseGenerating3D`
	`BaseSRFwithEllipsoidalHeight`: `CreateSurfaceCoordinate`, `GetSurfaceCoordinateValues`, `AssociateSurfaceCoordinate`, `PromoteSurfaceCoordinate`, `CreateLocalTangentSpaceEuclideanSRF`, `VerticalOffset`, `GeodesicDistance`, `GeodesicDistanceWithAzimuths`, `GeodesicDestination`, `EuclideanDistance`

Abstract method	`SetValidGeodeticRegion`
Semantics	This method creates a numeric interval for valid values of a surface geodetic coordinate-component. (See 8.3.2.4.) Given a geodetic coordinate-component, the last invocation of this method or the `SetExtendedValidGeodeticRegion` method determines the valid interval of the coordinate-component values. If an extended value interval has previously been set for a coordinate-component by the `SetExtendedValidGeodeticRegion` method, the extended interval limits shall be adjusted as necessary to contain the valid interval. The inputs `upper_bound` and `lower_bound` shall be in radians. The values of `lower_bound` and `upper_bound` are ignored if type is `UNBOUNDED`.

Inputs	`component_identifier:` `Integer;`
	`interval_type:` `Interval_Type;`
	`lower_bound:` `Long_Float;`
	`upper_bound:` `Long_Float;`

Outputs	none
Error conditions	`INVALID_INPUT` if a) the value of `component_identifier` is not 1 or 2, or b) the value of `type` is a semi-interval type, or c) the value of `type` is a bounded interval and the value of `lower_bound` or `upper_bound` does not satisfy the surface geodetic CS domain constraint or the values are equal.

Element	Specification
Abstract method	SetExtendedValidGeodeticRegion
Semantics	This method creates numeric intervals for both valid values and extended valid values of a surface geodetic coordinate-component. (See 8.3.2.4.) Given a coordinate-component, the last invocation of this method or the SetValidGeodeticRegion method determines the valid and extended valid intervals of the coordinate-component values. The inputs upper_bound, extended_upper_bound, lower_bound, and extended_lower_bound shall be in radians. The values of lower_bound, extended_lower_bound, upper_bound and extended_upper_bound are ignored if type is UNBOUNDED.
Inputs	component_identifier: Integer; interval_type: Interval_Type; extended_lower_bound: Long_Float; lower_bound: Long_Float; upper_bound: Long_Float; extended_upper_bound: Long_Float
Outputs	none
Error conditions	INVALID_INPUT if a) the value of component_identifier is not 1 or 2, or b) the value of type is a semi-interval type, or c) the value of type is a bounded interval and the value of lower_bound or upper_bound does not satisfy the surface geodetic CS domain constraint or the values are equal, or d) the interval determined by the values extended_lower_bound and extended_upper_bound does not contain the valid region.
Abstract method	GetValidGeodeticRegion
Semantics	This method returns the numeric interval for valid values of a surface geodetic coordinate-component that have been set in the last SetValidGeodeticRegion or SetExtendedValidGeodeticRegion method invocation for the same coordinate-component. If an interval has not been set for the coordinate-component, the output type shall be the Interval_Type UNBOUNDED. If the output type is UNBOUNDED, the output values of lower_bound and upper_bound shall be 0.0.
Inputs	component_identifier: Integer;
Outputs	interval_type: Interval_Type; lower_bound: Long_Float; upper_bound: Long_Float;
Error conditions	INVALID_INPUT if the value of component_identifier is not 1 or 2.

Element	Specification
Abstract method	GetExtendedValidGeodeticRegion
Semantics	This method returns the numeric intervals for valid and extended values of a surface geodetic coordinate-component that have been set in the last SetValidGeodeticRegion or SetExtendedValidGeodeticRegion method invocation for the same coordinate-component. If an interval has not been set for the coordinate-component, the output type shall be the Interval_Type UNBOUNDED. If SetExtendedValidGeodeticRegion has not been invoked for the coordinate-component, then the output extended_lower_bound shall equal the output lower_bound and the output extended_upper_bound shall equal the output upper_bound. If the output type is UNBOUNDED, the output values of lower_bound, extended_lower_bound, upper_bound and extended_upper_bound shall be 0.0.
Inputs	component_identifier: Integer;
Outputs	interval_type: Interval_Type; extended_lower_bound: Long_Float; lower_bound: Long_Float; upper_bound: Long_Float; extended_upper_bound: Long_Float
Error conditions	INVALID_INPUT if the value of component_identifier is not 1 or 2.
Abstract method	ConvergenceOfTheMeridian
Semantics	Outputs the Convergence of the Meridian in radians at a position on the surface of the ellipsoid RD.
Inputs	surface_coordinate: SurfaceCoordinate
Outputs	gamma: Long_Float
Error conditions	INVALID_SOURCE_COORDINATE if surface_coordinate is not in the accuracy domain of this SRF.
Abstract method	PointDistortion
Semantics	Outputs the point distortion at a position on the surface of the ellipsoid RD.
Inputs	surface_coordinate: SurfaceCoordinate
Outputs	distortion: Long_Float
Error conditions	INVALID_SOURCE_COORDINATE if surface_coordinate is not in the accuracy domain of this SRF.
Abstract method	MapAzimuth
Semantics	Outputs the map azimuth in radians at the point1_coordinate towards the point2_coordinate.
Inputs	point1_coordinate: SurfaceCoordinate; point2_coordinate: SurfaceCoordinate;
Outputs	azimuth: Long_Float
Error conditions	1) INVALID_POINT1_COORDINATE if point1_coordinate is not a valid surface coordinate in this SRF. 2) INVALID_POINT2_COORDINATE if point2_coordinate is not a valid surface coordinate in this SRF.

11.3.5.7 Orientation

The class `Orientation` is an abstract class that represents the orientation of one SRF with respect to another. It provides the following methods that are common to all orientation representations:

```
GetMatrix3x3,
GetAxisAngle,
GetEulerAnglesZXZ,
GetTaitBryanAngles,
GetQuaternion,
ComposeWith,
TransformVector.
```

Table 11.19 — Orientation

Element	Specification
Class	`Orientation`
Description	An abstract class that represents the orientation of one SRF with respect to another. It provides methods that are common to all orientation representations.
Superclass(es)	`LifeCycleObject`: `Create`, `Destroy`
Abstract method	`GetMatrix3x3`
Semantics	This method returns the representation of the `Orientation` object in the form of a 3x3 rotation matrix. (See 6.4.2.)
Inputs	none
Outputs	`matrix: Matrix_3x3_Parameters;`
Error conditions	none
Abstract method	`GetAxisAngle`
Semantics	This method returns the representation of the `Orientation` object in the form of a unit vector ***n*** (with components n_1, n_2, and n_3) and a rotation angle θ. (See 6.4.3.)
Inputs	none
Outputs	`axis_angle: Axis_Angle_Parameters;`
Error conditions	none
Abstract method	`GetEulerAnglesZXZ`
Semantics	This method returns the representation of the `Orientation` object in the form of three consecutive Euler rotation angles about the principal coordinate system axes. (See 6.4.4.3.) The first rotation, $\mathbf{\Omega}_z(\alpha)$, is about the z-axis, through angle α. The second rotation, $\mathbf{\Omega}_x(\beta)$, is about the (original) x-axis, through angle β. The third rotation, $\mathbf{\Omega}_z(\gamma)$, is again about the (original) z-axis, through angle γ.
Inputs	none
Outputs	`euler_angles_ZXZ: Euler_Angles_ZXZ_Parameters;`
Error conditions	none

Element	Specification
Abstract method	`GetTaitBryanAngles`
Semantics	This method returns the representation of the `Orientation` object in the form three consecutive Tait-Bryan rotation angles about the principal coordinate system axes. (See 6.4.4.4.) The first rotation, $\mathbf{\Omega}_x(\varphi)$, is about the x-axis, through angle φ. The second rotation, $\mathbf{\Omega}_y(\theta)$, is about the (original) y-axis, through angle θ. The third rotation, $\mathbf{\Omega}_z(\psi)$, is about the (original) z-axis, through angle ψ.
Inputs	none
Outputs	`tait_bryan: Tait_Bryan_Parameters;`
Error conditions	none
Abstract method	`GetQuaternion`
Semantics	This method returns the representation of the `Orientation` object in the form of a quaternion. (See 6.4.5.)
Inputs	none
Outputs	`quaternion: Quaternion_Parameters;`
Error conditions	none
Abstract method	`ComposeWith`
Semantics	This method composes two given `Orientation` objects and returns the resulting `Orientation` object. Thus, if S_1, S_2, and S_3 are three SRFs, $\texttt{Orientation}_{12}$ (i.e., the orientation of S_1 with respect to S_2) is the input orientation object, and $\texttt{Orientation}_{23}$ (i.e., the orientation of S_2 with respect to S_3) is this orientation object, then the result $\texttt{Orientation}_{13}$, is the orientation of S_1 with respect to S_3. (See 10.5.5.)
Inputs	`orientation_1_2: Orientation;`
Outputs	`orientation_1_3: Orientation;`
Error conditions	none
Abstract method	`TransformVector`
Semantics	This method transforms the representation of a three-dimensional vector from the source SRF of this `Orientation` object to the target SRF of this `Orientation` object. (See 10.5.6.)
Inputs	`source_vector: Vector_3D;`
Outputs	`target_vector: Vector_3D;`
Error conditions	none

11.3.6 SRF concrete subclasses of BaseSRF2D

11.3.6.1 Introduction

Concrete classes based on BaseSRF2D inherit all the methods of the following classes:

LifeCycleObject,
BaseSRF, and
BaseSRF2D

classes. They add class specific Create methods. In those cases for which the Create method has a class specific input data structure, a GetSRFParameters method for the class is also specified.

11.3.6.2 LocalSpaceRectangular2D

Table 11.20 — LocalSpaceRectangular2D

Element	Specification
Class	LocalSpaceRectangular2D
Description	An instance of this class corresponds to an instance of SRFT LOCAL_SPACE_RECTANGULAR_2D.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF2D: CreateCoordinate2D, GetCoordinate2Dvalues, ChangeCoordinate2DSRF, ChangeCoordinate2DSRFObject, ChangeCoordinate2DArraySRF, ChangeCoordinate2DArraySRFObject, EuclideanDistance, Generating2D, InverseGenerating2D
Method	Create
Semantics	Overrides the Create method of the superclass LifeCycleObject. Creates a LocalSpaceRectangular2D SRF corresponding to the input values. The ChangeCoordinate2DSRF method requires a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code parameters: LSR_2D_Parameters
Outputs	new_srf: LocalSpaceRectangular2D
Error conditions	1) INVALID_CODE if orm_code is not a valid ORM_Code, or the rt_code is not a valid RT_Code (or 0). 2) INVALID_INPUT if the ORM corresponding to ORM_Code is not valid for this SRFT, or the parameters are not valid for this SRF.
Method	GetSRFParameters
Semantics	This method outputs the SRF parameter data structure.
Inputs	none
Outputs	parameters: LSR_2D_Parameters
Error conditions	No additional error conditions. (See 11.3.2.)

11.3.6.3 LocalSpaceAzimuthal

Table 11.21 — LocalSpaceAzimuthal

Element	Specification
Class	LocalSpaceAzimuthal
Description	An instance of this class corresponds to an instance of SRFT LOCAL_SPACE_AZIMUTHAL_2D.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF2D: CreateCoordinate2D, GetCoordinate2Dvalues, ChangeCoordinate2DSRF, ChangeCoordinate2DSRFObject, ChangeCoordinate2DArraySRF, ChangeCoordinate2DArraySRFObject, EuclideanDistance, Generating2D, InverseGenerating2D
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a LocalSpaceAzimuthal SRF corresponding to the input values. The ChangeCoordinate2DSRF method requires a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code
Outputs	new_srf: LocalSpaceAzimuthal
Error conditions	INVALID_CODE if orm_code is not a valid ORM_Code, or the corresponding ORM is not valid for this SRFT, or the rt_code is not a valid RT_Code (or 0).

11.3.6.4 LocalSpacePolar

Table 11.22 — LocalSpacePolar

Element	Specification
Class	LocalSpacePolar
Description	An instance of this class corresponds to an instance of SRFT LOCAL_SPACE_POLAR_2D.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF2D: CreateCoordinate2D, GetCoordinate2Dvalues, ChangeCoordinate2DSRF, ChangeCoordinate2DSRFObject, ChangeCoordinate2DArraySRF, ChangeCoordinate2DArraySRFObject, EuclideanDistance, Generating2D, InverseGenerating2D
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a LocalSpacePolar SRF corresponding to the input ORM_Code parameter. The ChangeCoordinate2DSRF method require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.

Element	Specification
Inputs	orm_code: ORM_Code rt_code: RT_Code
Outputs	new_srf: LocalSpacePolar
Error conditions	INVALID_CODE if orm_code is not a valid ORM_Code, or the corresponding ORM is not valid for this SRFT, or the rt_code is not a valid RT_Code (or 0).

11.3.7 SRF concrete subclasses of BaseSRF3D

11.3.7.1 Introduction

Concrete classes based on BaseSRF3D inherit all the methods of the following classes:

LifeCycleObject,
BaseSRF, and
BaseSRF3D.

They add class specific Create methods. In those cases for which the Create method has a class specific input data structure, a GetSRFParameters method for the class is also specified.

11.3.7.2 Celestiocentric

Table 11.23 — Celestiocentric

Element	Specification
Class	Celestiocentric
Description	An instance of this class corresponds to an instance of SRFT CELESTIOCENTRIC.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a Celestiocentric SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.

Element	Specification
Inputs	orm_code: ORM_Code rt_code: RT_Code
Outputs	new_srf: Celestiocentric
Error conditions	INVALID_CODE if orm_code is not a valid ORM_Code, or the corresponding ORM is not valid for this SRFT, or the rt_code is not a valid RT_Code (or 0).

11.3.7.3 LocalSpaceRectangular3D

Table 11.24 — LocalSpaceRectangular3D

Element	Specification
Class	LocalSpaceRectangular3D
Description	An instance of this class corresponds to an instance of SRFT LOCAL_SPACE_RECTANGULAR_3D.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a LocalSpaceRectangular3D SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code parameters: LSR_3D_Parameters
Outputs	new_srf: LocalSpaceRectangular3D
Error conditions	1) INVALID_CODE if orm_code is not a valid ORM_Code, or the rt_code is not a valid RT_Code (or 0). 2) INVALID_INPUT if the ORM corresponding to ORM_Code is not valid for this SRFT, or the parameters are not valid for this SRF.
Method	GetSRFParameters
Semantics	This method outputs the SRF parameter data structure.
Inputs	none
Outputs	parameters: LSR_3D_Parameters
Error conditions	No additional error conditions. (See 11.3.2.)

11.3.7.4 LococentricEuclidean3D

Table 11.25 — LococentricEuclidean3D

Element	Specification
Class	LococentricEuclidean3D
Description	An instance of this class corresponds to an instance of SRFT LOCOCENTRIC_EUCLIDEAN_3D.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a Celestiocentric SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code parameters: LCE_3D_Parameters
Outputs	new_srf: LococentricEuclidean3D
Error conditions	1) INVALID_CODE if orm_code is not a valid ORM_Code, or the rt_code is not a valid RT_Code (or 0). 2) INVALID_INPUT if the ORM corresponding to ORM_Code is not valid for this SRFT, or the parameters are not valid for this SRF.
Method	GetSRFParameters
Semantics	This method outputs the SRF parameter data structure.
Inputs	none
Outputs	parameters: LCE_3D_Parameters
Error conditions	No additional error conditions. (See 11.3.2.)

11.3.7.5 Celestiomagnetic

Table 11.26 — Celestiomagnetic

Element	Specification
Class	`Celestiomagnetic`
Description	An instance of this class corresponds to an instance of SRFT CELESTIOMAGNETIC.

Superclass(es)	`LifeCycleObject`: `Destroy`
	`BaseSRF`: `GetORMCodes`, `GetSRFCodes`, `GetCSCode`
	`BaseSRF3D`: `CreateCoordinate3D`, `SetValidRegion`, `SetExtendedValidRegion`, `GetValidRegion`, `GetExtendedValidRegion`, `CreateDirection`, `GetCoordinate3Dvalues`, `GetDirectionValues`, `ChangeCoordinate3DSRF`, `ChangeCoordinate3DSRFObject`, `ChangeCoordinate3DArraySRF`, `ChangeCoordinate3DArraySRFObject`, `ChangeDirectionSRF`, `ChangeDirectionSRFObject`, `ChangeDirectionArraySRF`, `ChangeDirectionArraySRFObject`, `EuclideanDistance`, `CreateLococentricEuclidean3DSRF`, `Generating3D`, `InverseGenerating3D`

Method	`Create`
Semantics	Overrides the `Create` method on the superclass `LifeCycleObject`. Creates a `Celestiomagnetic` SRF corresponding to the input values. The `ChangeCoordinate3DSRF` and `ChangeDirectionSRF` methods of the output SRF object require a valid `rt_code` value, otherwise the `rt_code` value 0 (`UNSPECIFIED`) is permitted.
Inputs	`orm_code:` `ORM_Code` `rt_code:` `RT_Code`
Outputs	`new_srf:` `Celestiomagnetic`
Error conditions	`INVALID_CODE` if orm is not a valid `ORM_Code` or the corresponding ORM is not valid for this SRFT, or the `rt_code` is not a valid `RT_Code` (or 0).

11.3.7.6 EquatorialInertial

Table 11.27 — EquatorialInertial

Element	Specification
Class	`EquatorialInertial`
Description	An instance of this class corresponds to an instance of SRFT EQUATORIAL_INERTIAL.

Element	Specification
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates an EquatorialInertial SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code
Outputs	new_srf: EquatorialInertial
Error conditions	INVALID_CODE if orm_code is not a valid ORM_Code, or the corresponding ORM is not valid for this SRFT, or the rt_code is not a valid RT_Code (or 0).

11.3.7.7 SolarEcliptic

Table 11.28 — SolarEcliptic

Element	Specification
Class	SolarEcliptic
Description	An instance of this class corresponds to an instance of SRFT SOLAR_ECLIPTIC.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D

Element	Specification
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a SolarEcliptic SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code
Outputs	new_srf: SolarEcliptic
Error conditions	INVALID_CODE if orm_code is not a valid ORM_Code, or the corresponding ORM is not valid for this SRFT, or the rt_code is not a valid RT_Code (or 0).

11.3.7.8 SolarEquatorial

Table 11.29 — SolarEquatorial

Element	Specification
Class	SolarEquatorial
Description	An instance of this class corresponds to an instance of SRFT SOLAR_EQUATORIAL.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a SolarEquatorial SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code
Outputs	new_srf: SolarEquatorial
Error conditions	INVALID_CODE if orm_code is not a valid ORM_Code, or the corresponding ORM is not valid for this SRFT, or the rt_code is not a valid RT_Code (or 0).

11.3.7.9 SolarMagneticEcliptic

Table 11.30 — SolarMagneticEcliptic

Element	Specification
Class	SolarMagneticEcliptic
Description	An instance of this class corresponds to an instance of SRFT SOLAR_MAGNETIC_ECLIPTIC.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a SolarMagneticEcliptic SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code
Outputs	new_srf: SolarMagneticEcliptic
Error conditions	INVALID_CODE if orm_code is not a valid ORM_Code, or the corresponding ORM is not valid for this SRFT, or the rt_code is not a valid RT_Code (or 0).

11.3.7.10 SolarMagneticDipole

Table 11.31 — SolarMagneticDipole

Element	Specification
Class	SolarMagneticDipole
Description	An instance of this class corresponds to an instance of SRFT SOLAR_MAGNETIC_DIPOLE.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a SolarMagneticDipole SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code
Outputs	new_srf: SolarMagneticDipole
Error conditions	INVALID_CODE if orm_code is not a valid ORM_Code, or the corresponding ORM is not valid for this SRFT, or the rt_code is not a valid RT_Code (or 0).

11.3.7.11 HeliosphericAriesEcliptic

Table 11.32 — HeliosphericAriesEcliptic

Element	Specification
Class	HeliosphericAriesEcliptic
Description	An instance of this class corresponds to an instance of SRFT HELIOSPHERIC_ARIES_ECLIPTIC.

Element	Specification
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a HeliosphericAriesEcliptic SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code
Outputs	new_srf: HeliosphericAriesEcliptic
Error conditions	INVALID_CODE if orm_code is not a valid ORM_Code, or the corresponding ORM is not valid for this SRFT, or the rt_code is not a valid RT_Code (or 0).

11.3.7.12 HeliosphericEarthEcliptic

Table 11.33 — HeliosphericEarthEcliptic

Element	Specification
Class	HeliosphericEarthEcliptic
Description	An instance of this class corresponds to an instance of SRFT HELIOSPHERIC_EARTH_ECLIPTIC.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D

Element	Specification
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a HeliosphericEarthEcliptic SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code
Outputs	new_srf: HeliosphericEarthEcliptic
Error conditions	INVALID_CODE if orm_code is not a valid ORM_Code, or the corresponding ORM is not valid for this SRFT, or the rt_code is not a valid RT_Code (or 0).

11.3.7.13 HeliosphericEarthEquatorial

Table 11.34 — HeliosphericEarthEquatorial

Element	Specification
Class	HeliosphericEarthEquatorial
Description	An instance of this class corresponds to an instance of SRFT HELIOSPHERIC_EARTH_EQUATORIAL.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a HeliosphericEarthEquatorial SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code
Outputs	new_srf: HeliosphericEarthEquatorial
Error conditions	INVALID_CODE if orm_code is not a valid ORM_Code, or the corresponding ORM is not valid for this SRFT, or the rt_code is not a valid RT_Code (or 0).

11.3.8 SRF concrete subclasses of BaseSRFwithTangentPlaneSurface

11.3.8.1 Introduction

Concrete classes based on BaseSRFwithTangentPlaneSurface inherit all the methods of the following classes:

LifeCycleObject,
BaseSRF, BaseSRF3D, and
BaseSRFwithTangentPlaneSurface.

They add class specific Create and GetSRFParameters methods.

11.3.8.2 LocalTangentSpaceEuclidean

Table 11.35 — LocalTangentSpaceEuclidean

Element	Specification
Class	LocalTangentSpaceEuclidean
Description	An instance of this class corresponds to an instance of SRFT LOCAL_TANGENT_SPACE_EUCLIDEAN.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D BaseSRFwithTangentPlaneSurface: CreateSurfaceCoordinate, GetSurfaceCoordinateValues, AssociateSurfaceCoordinate, PromoteSurfaceCoordinate, EuclideanDistance
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a LocalTangentEuclidean SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code parameters: LTSE_Parameters
Outputs	new_srf: LocalTangentSpaceEuclidean

Element	Specification
Error conditions	1) INVALID_CODE if orm_code is not a valid ORM_Code, or the rt_code is not a valid RT_Code (or 0). 2) INVALID_INPUT if the ORM corresponding to ORM_Code is not valid for this SRFT, or the parameters are not valid for this SRF.
Method	GetSRFParameters
Semantics	This method outputs the SRF parameters.
Inputs	none
Outputs	parameters: LTSE_Parameters
Error conditions	No additional error conditions. (See 11.3.2.)

11.3.8.3 LocalTangentSpaceAzimuthalSpherical

Table 11.36 — LocalTangentSpaceAzimuthalSpherical

Element	Specification
Class	LocalTangentSpaceAzimuthalSpherical
Description	An instance of this class corresponds to an instance of SRFT LOCAL_TANGENT_SPACE_AZIMUTHAL_SPHERICAL.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D BaseSRFwithTangentPlaneSurface: CreateSurfaceCoordinate, GetSurfaceCoordinateValues, AssociateSurfaceCoordinate, PromoteSurfaceCoordinate, EuclideanDistance
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a LocalTangentSpaceAzimuthalSpherical SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code parameters: Local_Tangent_Parameters
Outputs	new_srf: LocalTangentSpaceAzimuthalSpherical

Element	Specification
Error conditions	1) INVALID_CODE if orm_code is not a valid ORM_Code, or the rt_code is not a valid RT_Code (or 0). 2) INVALID_INPUT if the ORM corresponding to ORM_Code is not valid for this SRFT, or the parameters are not valid for this SRF.
Method	GetSRFParameters
Semantics	This method outputs the SRF parameters.
Inputs	none
Outputs	parameters: Local_Tangent_Parameters
Error conditions	No additional error conditions. (See 11.3.2.)

11.3.8.4 LocalTangentSpaceCylindrical

Table 11.37 — LocalTangentSpaceCylindrical

Element	Specification
Class	LocalTangentSpaceCylindrical
Description	An instance of this class corresponds to an instance of SRFT LOCAL_TANGENT_SPACE_CYLINDRICAL.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D BaseSRFwithTangentPlaneSurface: CreateSurfaceCoordinate, GetSurfaceCoordinateValues, AssociateSurfaceCoordinate, PromoteSurfaceCoordinate, EuclideanDistance
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a LocalTangentSpaceCylindrical SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code parameters: Local_Tangent_Parameters
Outputs	new_srf: LocalTangentSpaceCylindrical

Element	Specification
Error conditions	1) INVALID_CODE if orm_code is not a valid ORM_Code, or the rt_code is not a valid RT_Code (or 0). 2) INVALID_INPUT if the ORM corresponding to ORM_Code is not valid for this SRFT, or the parameters are not valid for this SRF.
Method	GetSRFParameters
Semantics	This method outputs the SRF parameters.
Inputs	none
Outputs	parameters: Local_Tangent_Parameters
Error conditions	No additional error conditions. (See 11.3.2.)

11.3.9 SRF concrete subclass of BaseSRFwithEllipsoidalHeight

11.3.9.1 Introduction

Concrete classes based on BaseSRFwithEllipsoidalHeight inherit all the methods of the LifeCycleObject, BaseSRF, BaseSRF3D, and BaseSRFwithEllipsoidalHeight.

11.3.9.2 Celestiodetic

Table 11.38 — Celestiodetic

Element	Specification
Class	Celestiodetic
Description	An instance of this class corresponds to an instance of SRFT CELESTIODETIC.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D BaseSRFwithEllipsoidalHeight: CreateSurfaceCoordinate, GetSurfaceCoordinateValues, AssociateSurfaceCoordinate, PromoteSurfaceCoordinate, CreateLocalTangentSpaceEuclideanSRF, VerticalOffset, GeodesicDistance, GeodesicDistanceWithAzimuths, GeodesicDestination, EuclideanDistance

Element	Specification
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a Celestiodetic SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code
Outputs	new_srf: Celestiodetic
Error conditions	INVALID_CODE if orm_code is not a valid ORM_Code, or the corresponding ORM is not valid for this SRFT, or the rt_code is not a valid RT_Code (or 0).

11.3.9.3 Planetodetic

Table 11.39 — Planetodetic

Element	Specification
Class	Planetodetic
Description	An instance of this class corresponds to an instance of SRFT PLANETODETIC.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D BaseSRFwithEllipsoidalHeight: CreateSurfaceCoordinate, GetSurfaceCoordinateValues, AssociateSurfaceCoordinate, PromoteSurfaceCoordinate, CreateLocalTangentSpaceEuclideanSRF, VerticalOffset, GeodesicDistance, GeodesicDistanceWithAzimuths, GeodesicDestination, EuclideanDistance
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a Planetodetic SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code

Element	Specification
Outputs	new_srf: Planetodetic
Error conditions	INVALID_CODE if orm_code is not a valid ORM_Code, or the corresponding ORM is not valid for this SRFT, or the rt_code is not a valid RT_Code (or 0).

11.3.10 SRF concrete subclasses of BaseSRFMapProjection

11.3.10.1 Introduction

Concrete classes based on BaseSRFMapProjection inherit all the methods of the following classes:

LifeCycleObject,
BaseSRF,
BaseSRF3D,
BaseSRFwithEllipsoidalHeight, and
BaseSRFMapProjection.

They add class specific Create and GetSRFParameters methods.

11.3.10.2 Mercator

Table 11.40 — Mercator

Element	Specification
Class	Mercator
Description	An instance of this class corresponds to an instance of SRFT MERCATOR.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D BaseSRFwithEllipsoidalHeight: CreateSurfaceCoordinate, GetSurfaceCoordinateValues, AssociateSurfaceCoordinate, PromoteSurfaceCoordinate, CreateLocalTangentSpaceEuclideanSRF, VerticalOffset, GeodesicDistance, GeodesicDistanceWithAzimuths, GeodesicDestination, EuclideanDistance BaseSRFMapProjection: SetValidGeodeticRegion, SetExtendedValidGeodeticRegion, GetValidGeodeticRegion, GetExtendedValidGeodeticRegion, ConvergenceOfTheMeridian, PointDistortion, MapAzimuth

Element	Specification
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a Mercator SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code parameters: M_Parameters
Outputs	new_srf: Mercator
Error conditions	1) INVALID_CODE if orm_code is not a valid ORM_Code, or the rt_code is not a valid RT_Code (or 0). 2) INVALID_INPUT if the ORM corresponding to ORM_Code is not valid for this SRFT, or the parameters are not valid for this SRF.
Method	GetSRFParameters
Semantics	This method outputs the SRF parameters.
Inputs	none
Outputs	parameters: M_Parameters
Error conditions	No additional error conditions. (See 11.3.2.)

11.3.10.3 ObliqueMercatorSpherical

Table 11.41 — ObliqueMercatorSpherical

Element	Specification
Class	ObliqueMercatorSpherical
Description	An instance of this class corresponds to an instance of SRFT OBLIQUE_MERCATOR_SPHERICAL.

Element	Specification
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D BaseSRFwithEllipsoidalHeight: CreateSurfaceCoordinate, GetSurfaceCoordinateValues, AssociateSurfaceCoordinate, PromoteSurfaceCoordinate, CreateLocalTangentSpaceEuclideanSRF, VerticalOffset, GeodesicDistance, GeodesicDistanceWithAzimuths, GeodesicDestination, EuclideanDistance BaseSRFMapProjection: SetValidGeodeticRegion, SetExtendedValidGeodeticRegion, GetValidGeodeticRegion, GetExtendedValidGeodeticRegion, ConvergenceOfTheMeridian, PointDistortion, MapAzimuth
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates an ObliqueMercatorSpherical SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code parameters: Oblique_Mercator_Parameters
Outputs	new_srf: ObliqueMercatorSpherical
Error conditions	1) INVALID_CODE if orm_code is not a valid ORM_Code, or the rt_code is not a valid RT_Code (or 0). 2) INVALID_INPUT if the ORM corresponding to ORM_Code is not valid for this SRFT, or the parameters are not valid for this SRF.
Method	GetSRFParameters
Semantics	This method outputs the SRF parameters.
Inputs	none
Outputs	parameters: Oblique_Mercator_Parameters
Error conditions	No additional error conditions. (See 11.3.2.)

11.3.10.4 TransverseMercator

Table 11.42 — TransverseMercator

Element	Specification
Class	TransverseMercator
Description	An instance of this class corresponds to an instance of SRFT TRANSVERSE_MERCATOR.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D BaseSRFwithEllipsoidalHeight: CreateSurfaceCoordinate, GetSurfaceCoordinateValues, AssociateSurfaceCoordinate, PromoteSurfaceCoordinate, CreateLocalTangentSpaceEuclideanSRF, VerticalOffset, GeodesicDistance, GeodesicDistanceWithAzimuths, GeodesicDestination, EuclideanDistance BaseSRFMapProjection: SetValidGeodeticRegion, SetExtendedValidGeodeticRegion, GetValidGeodeticRegion, GetExtendedValidGeodeticRegion, ConvergenceOfTheMeridian, PointDistortion, MapAzimuth
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a TransverseMercator SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code parameters: TM_Parameters
Outputs	new_srf: TransverseMercator
Error conditions	1) INVALID_CODE if orm_code is not a valid ORM_Code, or the rt_code is not a valid RT_Code (or 0). 2) INVALID_INPUT if the ORM corresponding to ORM_Code is not valid for this SRFT, or the parameters are not valid for this SRF.
Method	GetSRFParameters
Semantics	This method outputs the SRF parameters.
Inputs	none
Outputs	parameters: TM_Parameters

Element	Specification
Error conditions	No additional error conditions. (See 11.3.2.)

11.3.10.5 LambertConformalConic

Table 11.43 — LambertConformalConic

Element	Specification
Class	LambertConformalConic
Description	An instance of this class corresponds to an instance of SRFT LAMBERT_CONFORMAL_CONIC.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D BaseSRFwithEllipsoidalHeight: CreateSurfaceCoordinate, GetSurfaceCoordinateValues, AssociateSurfaceCoordinate, PromoteSurfaceCoordinate, CreateLocalTangentSpaceEuclideanSRF, VerticalOffset, GeodesicDistance, GeodesicDistanceWithAzimuths, GeodesicDestination, EuclideanDistance BaseSRFMapProjection: SetValidGeodeticRegion, SetExtendedValidGeodeticRegion, GetValidGeodeticRegion, GetExtendedValidGeodeticRegion, ConvergenceOfTheMeridian, PointDistortion, MapAzimuth
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a LambertConformalConic SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code parameters: LCC_Parameters
Outputs	new_srf: LambertConformalConic

Element	Specification
Error conditions	1) INVALID_CODE if orm_code is not a valid ORM_Code, or the rt_code is not a valid RT_Code (or 0). 2) INVALID_INPUT if the ORM corresponding to ORM_Code is not valid for this SRFT, or the parameters are not valid for this SRF.
Method	GetSRFParameters
Semantics	This method outputs the SRF parameters.
Inputs	none
Outputs	parameters: LCC_Parameters
Error conditions	No additional error conditions. (See 11.3.2.)

11.3.10.6 PolarStereographic

Table 11.44 — PolarStereographic

Element	Specification
Class	PolarStereographic
Description	An instance of this class corresponds to an instance of SRFT POLAR_STEREOGRAPHIC.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D BaseSRFwithEllipsoidalHeight: CreateSurfaceCoordinate, GetSurfaceCoordinateValues, AssociateSurfaceCoordinate, PromoteSurfaceCoordinate, CreateLocalTangentSpaceEuclideanSRF, VerticalOffset, GeodesicDistance, GeodesicDistanceWithAzimuths, GeodesicDestination, EuclideanDistance BaseSRFMapProjection: SetValidGeodeticRegion, SetExtendedValidGeodeticRegion, GetValidGeodeticRegion, GetExtendedValidGeodeticRegion, ConvergenceOfTheMeridian, PointDistortion, MapAzimuth
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates a PolarStereographic SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code

Element	Specification
	value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code parameters: PS_Parameters
Outputs	new_srf: PolarStereographic
Error conditions	1) INVALID_CODE if orm_code is not a valid ORM_Code, or the rt_code is not a valid RT_Code (or 0). 2) INVALID_INPUT if the ORM corresponding to ORM_Code is not valid for this SRFT, or the parameters are not valid for this SRF.
Method	GetSRFParameters
Semantics	This method outputs the SRF parameters.
Inputs	none
Outputs	parameters: PS_Parameters
Error conditions	No additional error conditions. (See 11.3.2.)

11.3.10.7 EquidistantCylindrical

Table 11.45 — EquidistantCylindrical

Element	Specification
Class	EquidistantCylindrical
Description	An instance of this class corresponds to an instance of SRFT EQUIDISTANT_CYLINDRICAL.
Superclass(es)	LifeCycleObject: Destroy BaseSRF: GetORMCodes, GetSRFCodes, GetCSCode BaseSRF3D: CreateCoordinate3D, SetValidRegion, SetExtendedValidRegion, GetValidRegion, GetExtendedValidRegion, CreateDirection, GetCoordinate3Dvalues, GetDirectionValues, ChangeCoordinate3DSRF, ChangeCoordinate3DSRFObject, ChangeCoordinate3DArraySRF, ChangeCoordinate3DArraySRFObject, ChangeDirectionSRF, ChangeDirectionSRFObject, ChangeDirectionArraySRF, ChangeDirectionArraySRFObject, EuclideanDistance, CreateLococentricEuclidean3DSRF, Generating3D, InverseGenerating3D BaseSRFwithEllipsoidalHeight: CreateSurfaceCoordinate, GetSurfaceCoordinateValues, AssociateSurfaceCoordinate, PromoteSurfaceCoordinate, CreateLocalTangentSpaceEuclideanSRF, VerticalOffset, GeodesicDistance, GeodesicDistanceWithAzimuths, GeodesicDestination, EuclideanDistance BaseSRFMapProjection: SetValidGeodeticRegion, SetExtendedValidGeodeticRegion, GetValidGeodeticRegion, GetExtendedValidGeodeticRegion, ConvergenceOfTheMeridian, PointDistortion, MapAzimuth

Element	Specification
Method	Create
Semantics	Overrides the Create method on the superclass LifeCycleObject. Creates an EquidistantCylindrical SRF corresponding to the input values. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	orm_code: ORM_Code rt_code: RT_Code parameters: EC_Parameters
Outputs	new_srf: EquidistantCylindrical
Error conditions	1) INVALID_CODE if orm_code is not a valid ORM_Code, or the rt_code is not a valid RT_Code (or 0). 2) INVALID_INPUT if the ORM corresponding to ORM_Code is not valid for this SRFT, or the parameters are not valid for this SRF.
Method	GetSRFParameters
Semantics	This method outputs the SRF parameters.
Inputs	none
Outputs	parameters: EC_Parameters
Error conditions	No additional error conditions. (See 11.3.2.)

11.3.11 Concrete subclasses of Orientation

11.3.11.1 Introduction

Concrete classes based on Orientation inherit all of its methods. They add class specific Create, Set, and Get functions.

11.3.11.2 OrientationAxisAngle

Table 11.46 — OrientationAxisAngle

Element	Specification
Class	OrientationAxisAngle
Description	An instance of this class corresponds to an orientation using an axis-angle representation.
Superclass(es)	LifeCycleObject: Destroy Orientation: GetMatrix3x3, GetAxisAngle, GetEulerAnglesZXZ, GetTaitBryanAngles, GetQuaternion, ComposeWith, TransformVector
Method	Create
Semantics	Creates an OrientationAxisAngle instance according to the input values.
Inputs	axis: Vector_3D angle: Long_Float

Element	Specification
Outputs	`new_orientation: OrientationAxisAngle`
Error conditions	none
Method	`Set`
Semantics	This method sets the parameters of the `OrientationAxisAngle` instance.
Inputs	`parameters: Axis_Angle_Parameters`
Outputs	none
Error conditions	none
Method	`Get`
Semantics	This method outputs the parameters of the `OrientationAxisAngle` instance.
Inputs	none
Outputs	`parameters: Axis_Angle_Parameters`
Error conditions	none

11.3.11.3 OrientationEulerAnglesZXZ

Table 11.47 — OrientationEulerAnglesZXZ

Element	Specification
Class	`OrientationEulerAnglesZXZ`
Description	An instance of this class corresponds to an orientation using an Euler angle ZXZ representation.
Superclass(es)	`LifeCycleObject: Destroy` `Orientation: GetMatrix3x3, GetAxisAngle, GetEulerAnglesZXZ, GetTaitBryanAngles, GetQuaternion, ComposeWith, TransformVector`
Method	`Create`
Semantics	Creates an `OrientationEulerAnglesZXZ` instance according to the input values.

Inputs	`spin: Long_Float;`
	`nutation: Long_Float;`
	`precession: Long_Float;`

Outputs	`new_orientation: OrientationEulerAnglesZXZ`
Error conditions	none
Method	`Set`
Semantics	This method sets the parameters of the `OrientationEulerAnglesZXZ` instance.
Inputs	`parameters: Euler_Angles_ZXZ_Parameters`
Outputs	none
Error conditions	none

Element	Specification
Method	Get
Semantics	This method outputs the parameters of the OrientationEulerAnglesZXZ instance.
Inputs	none
Outputs	parameters: Euler_Angles_ZXZ_Parameters
Error conditions	none

11.3.11.4 OrientationTaitBryanAngles

Table 11.48 — OrientationTaitBryanAngles

Element	Specification
Class	OrientationTaitBryanAngles
Description	An instance of this class corresponds to an orientation with a Tait-Bryan angles representation.
Superclass(es)	LifeCycleObject: Destroy Orientation: GetMatrix3x3, GetAxisAngle, GetEulerAnglesZXZ, GetTaitBryanAngles, GetQuaternion, ComposeWith, TransformVector
Method	Create
Semantics	Creates an OrientationTaitBryanAngles instance according to the input values.
Inputs	roll: Long_Float; pitch: Long_Float; yaw: Long_Float;
Outputs	new_orientation: OrientationTaitBryanAngles
Error conditions	none
Method	Set
Semantics	This method sets the parameters of the OrientationTaitBryanAngles instance.
Inputs	parameters: Tait_Bryan_Parameters
Outputs	none
Error conditions	none
Method	Get
Semantics	This method outputs the parameters of the OrientationTaitBryanAngles instance.
Inputs	none
Outputs	parameters: Tait_Bryan_Parameters
Error conditions	none

11.3.11.5 OrientationMatrix

Table 11.49 — OrientationMatrix

Element	Specification
Class	OrientationMatrix
Description	An instance of this class corresponds to an orientation with a 3x3 rotation matrix representation.
Superclass(es)	LifeCycleObject: Destroy Orientation: GetMatrix3x3, GetAxisAngle, GetEulerAnglesZXZ, GetTaitBryanAngles, GetQuaternion, ComposeWith, TransformVector
Method	Create
Semantics	Creates an OrientationMatrix instance according to the input values.
Inputs	m: Matrix_3x3;
Outputs	new_orientation: OrientationMatrix
Error conditions	none
Method	Set
Semantics	This method sets the parameters of the OrientationMatrix instance.
Inputs	parameters: Matrix_3x3_Parameters
Outputs	none
Error conditions	none
Method	Get
Semantics	This method outputs the parameters of the OrientationMatrix instance.
Inputs	none
Outputs	parameters: Matrix_3x3_Parameters
Error conditions	none

11.3.11.6 OrientationQuaternion

Table 11.50 — OrientationQuaternion

Element	Specification
Class	OrientationQuaternion
Description	An instance of this class corresponds to an orientation with a quaternion representation.
Superclass(es)	LifeCycleObject: Destroy Orientation: GetMatrix3x3, GetAxisAngle, GetEulerAnglesZXZ, GetTaitBryanAngles, GetQuaternion, ComposeWith, TransformVector
Method	Create
Semantics	Creates an OrientationQuaternion instance according to the input values.

Element	Specification
Inputs	e0: Long_Float; e1: Long_Float; e2: Long_Float; e3: Long_Float;
Outputs	new_orientation: OrientationQuaternion
Error conditions	none
Method	Set
Semantics	This method sets the parameters of the OrientationQuaternion instance.
Inputs	parameters: Quaternion_Parameters
Outputs	none
Error conditions	none
Method	Get
Semantics	This method outputs the parameters of the OrientationQuaternion instance.
Inputs	none
Outputs	parameters: Quaternion_Parameters
Error conditions	none

11.4 Standard SRFs

This subclause defines a function that creates an SRF object that corresponds to one of the standard SRFs specified either in 8.6 or by registration (see 13.3.9). A standard SRF object differs from other instances of its SRFT class in that the GetSRFCodes method of the object outputs the corresponding SRF_Code. The function, CreateStandardSRF, specified in Table 11.51 creates and outputs a standard SRF object from an SRF_Code input. Each output SRF object corresponds to an entry in Table 8.31.

Table 11.51 — CreateStandardSRF

Element	Specification
Function	CreateStandardSRF
Semantics	The input SRF_Code determines a corresponding SRF entry in Table 8.31. The function creates and outputs an SRF object corresponding to the SRFT and template parameters specified in the SRF table entry. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	srf_code: SRF_Code rt_code: RT_Code
Outputs	new_srf: BaseSRF
Error conditions	1. INVALID_CODE if the srf_code is not a valid SRF_Code or the rt_code is not a valid RT_Code (or 0). 2. CREATION_FAILURE if the Create method of the corresponding SRF class fails.

EXAMPLE CreateStandardSRF with srf_code = 4 and an rt_code, produces as output an SRF object corresponding to SRF GEOCENTRIC_WGS_1984.

11.5 SRF set classes

This subclause defines a function that creates an SRF object that corresponds to a member of one of the standard SRF sets specified in 8.7. An SRF set member object differs from other instances of its SRFT class in that the GetSRFCodes method of the object outputs the corresponding SRFS member information in SRFS_Code_Info. The function CreateSRFSetMember creates and outputs an SRF object in the SRF set corresponding to the input SRFS_Code_Info and ORM_Code.

Table 11.52 — CreateSRFSetMember

Element	Specification
Function	CreateSRFSetMember
Semantics	The input SRFS_Code_Info and ORM_Code determine a corresponding member of a SRF set. The function creates and outputs an SRF object corresponding to that SRF Set member. The ChangeCoordinate3DSRF and ChangeDirectionSRF methods of the output SRF object require a valid rt_code value, otherwise the rt_code value 0 (UNSPECIFIED) is permitted.
Inputs	srfs_code_info: SRFS_Code_Info orm_code: ORM_Code rt_code: RT_Code
Outputs	new_srf: BaseSRF
Error conditions	1) INVALID_CODE if orm_code is not a valid ORM_Code, or the rt_code is not a valid RT_Code (or 0). 2) INVALID_INPUT if the srfs_code_info is not a valid SRFS_Code_Info, or the SRFS_Code_Info selector is of value SRFS_UNSPECIFIED, or the orm_code is not valid for the SRF set. 3) CREATION_FAILURE if the Create method of the corresponding SRF class fails.

EXAMPLE CreateSRFSetMember with parameters set_code_info.srfs_code = SRFS_ALABAMA_SPCS, set_code_info.SRFSM_alabama_spcs = 1, an orm_code and an rt_code, produces as output an SRF object corresponding to SRFS member ALABAMA_SPCS zone 1.

11.6 Implementation support query functions

Two query functions are provided to indicate the support of an API implementation for subsets of the SRM as may be defined by a profile. (see Clause 12 and 14.2). The function QuerySRFSupport indicates which SRF classes are supported. The function QueryORMSupport indicates which ORMs or ORM and RT combinations are supported.

Table 11.53 — QuerySRFTSupport

Element	Specification
Function	QuerySRFSupport
Semantics	If the implementation supports the full functionality and all the associated data types of the SRF class indicated by the input srft_code, then the output parameter supported is set to the Boolean value true. Otherwise, supported is set to the Boolean value false.
Inputs	srft_code: SRFT_Code
Outputs	supported: Boolean
Error conditions	INVALID_CODE if the srft_code is not an allowed value for an SRFT_Code.

Table 11.54 — QueryORMSupport

Element	Specification
Function	QueryORMSupport
Semantics	If the implementation supports the parameter values and all the associated data types of the ORM and the RT indicated by the inputs orm_code, and rt_code, then the output supported is set to the Boolean value true. Otherwise, the output supported is set to the Boolean value false. The rt_code value 0 (UNSPECIFIED) is permitted. In that case, the output supported is set to the Boolean value true only if the implementation supports the parameter values and all the associated data types of the ORM indicated by the input orm_code.
Inputs	orm_code: ORM_Code rt_code: RT_Code
Outputs	supported: Boolean
Error conditions	INVALID_CODE if orm_code is not a valid ORM_Code, or the rt_code is not a valid RT_Code (or 0).

11.7 Object inheritance hierarchy

The object inheritance hierarchy is summarized in Figure 11.1.

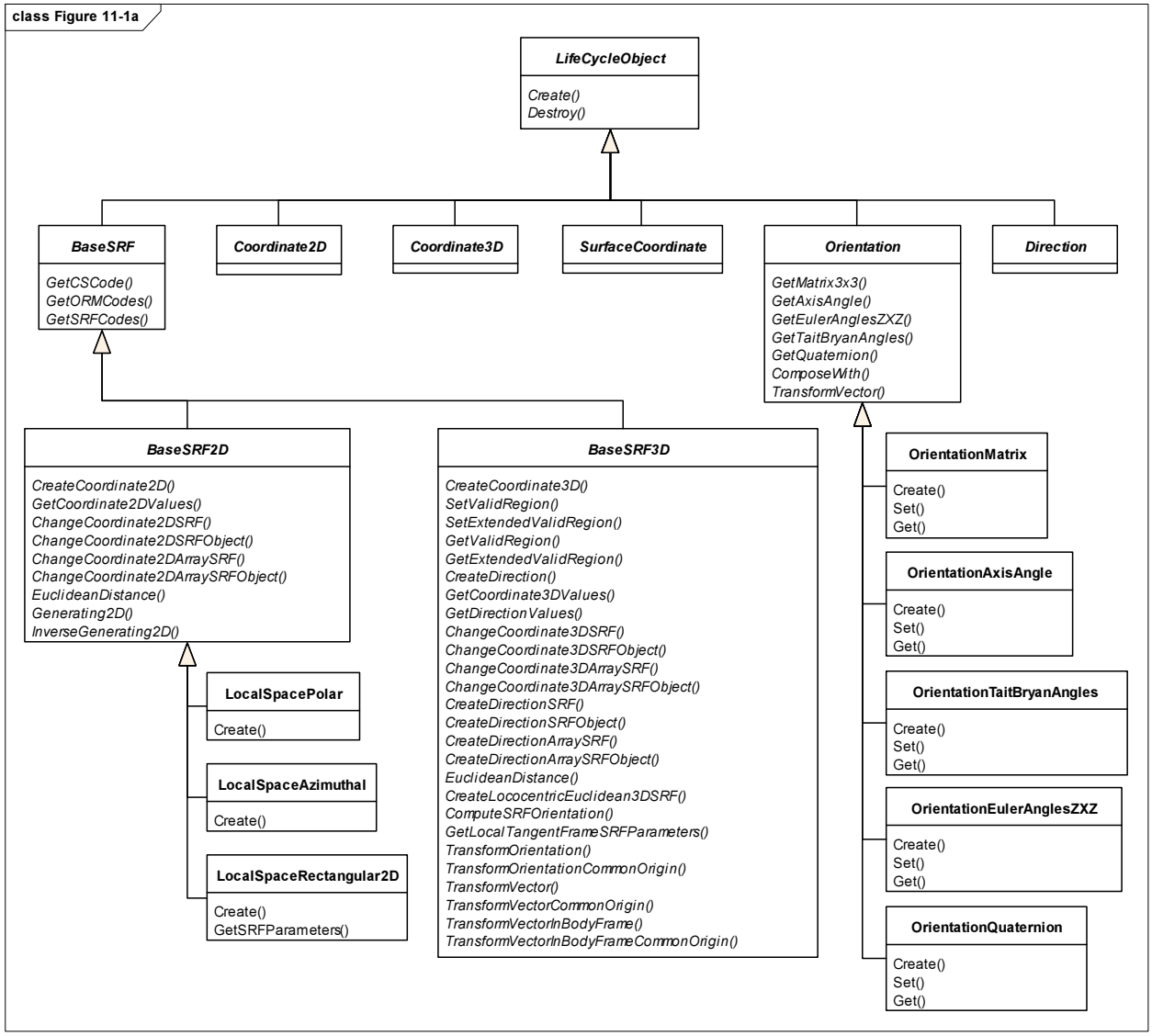

Figure 11.1 — Object inheritance hierarchy

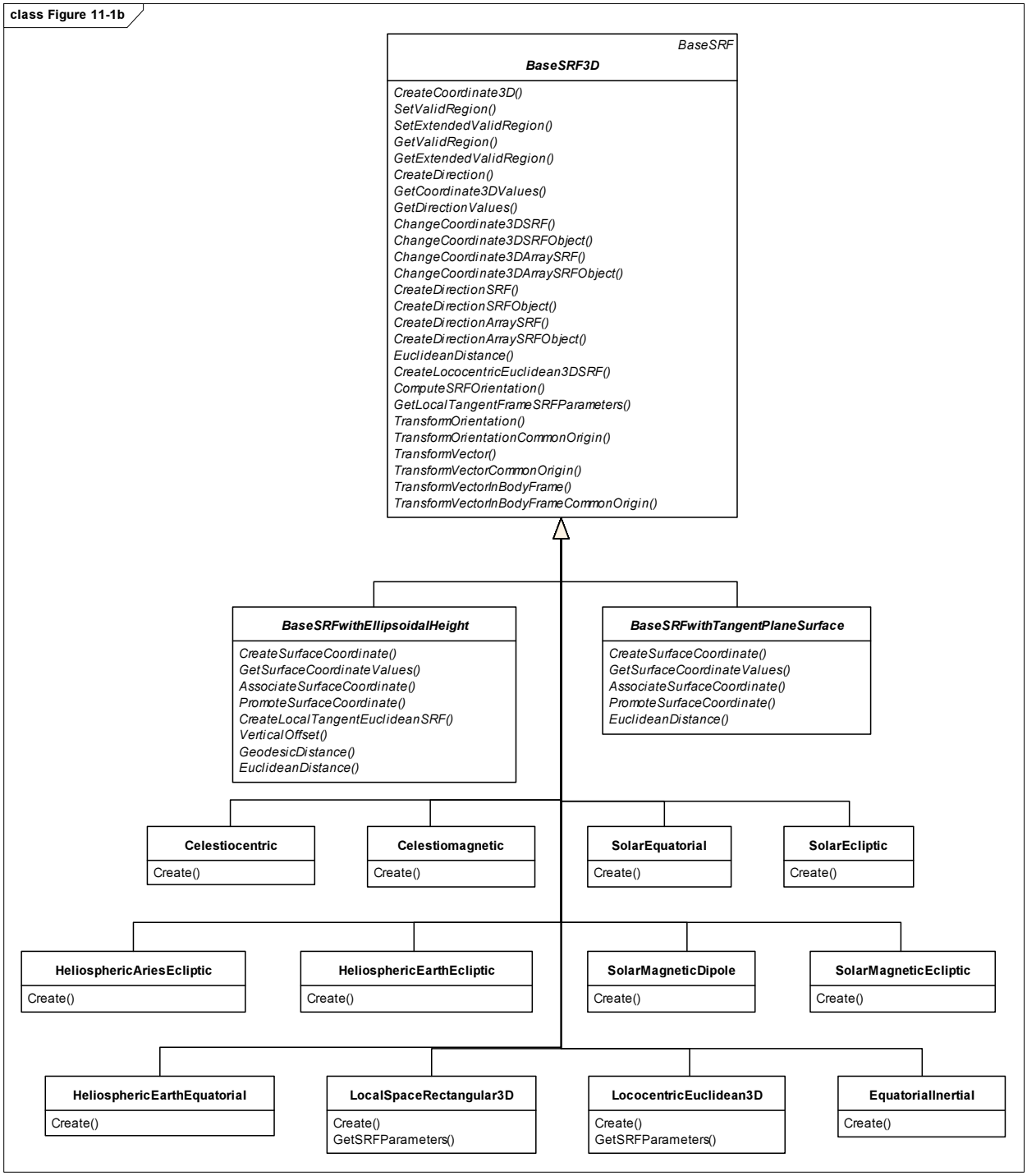

Figure 11.1 — Object inheritance hierarchy (continued)

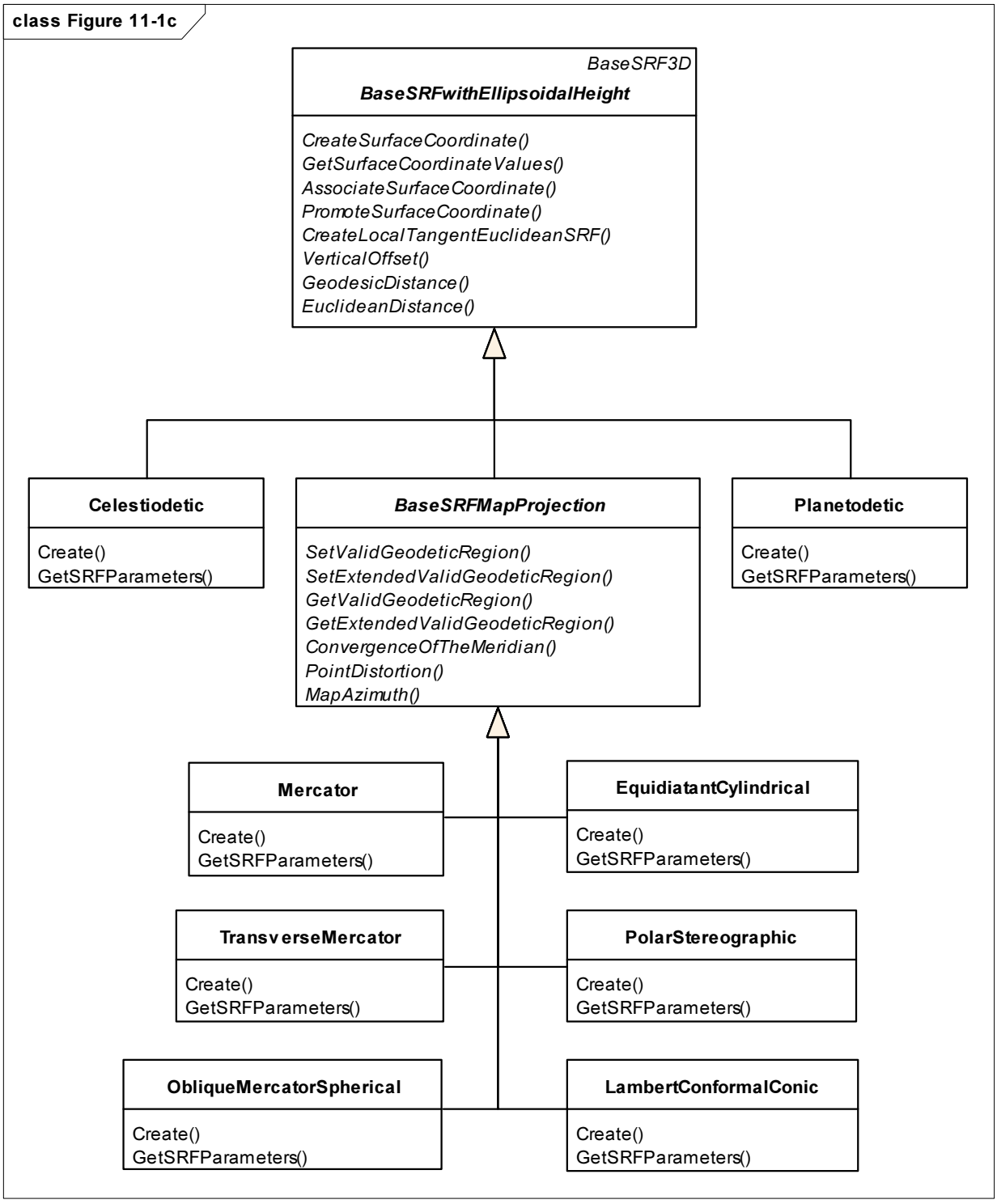

Figure 11.1 — Object inheritance hierarchy (continued)

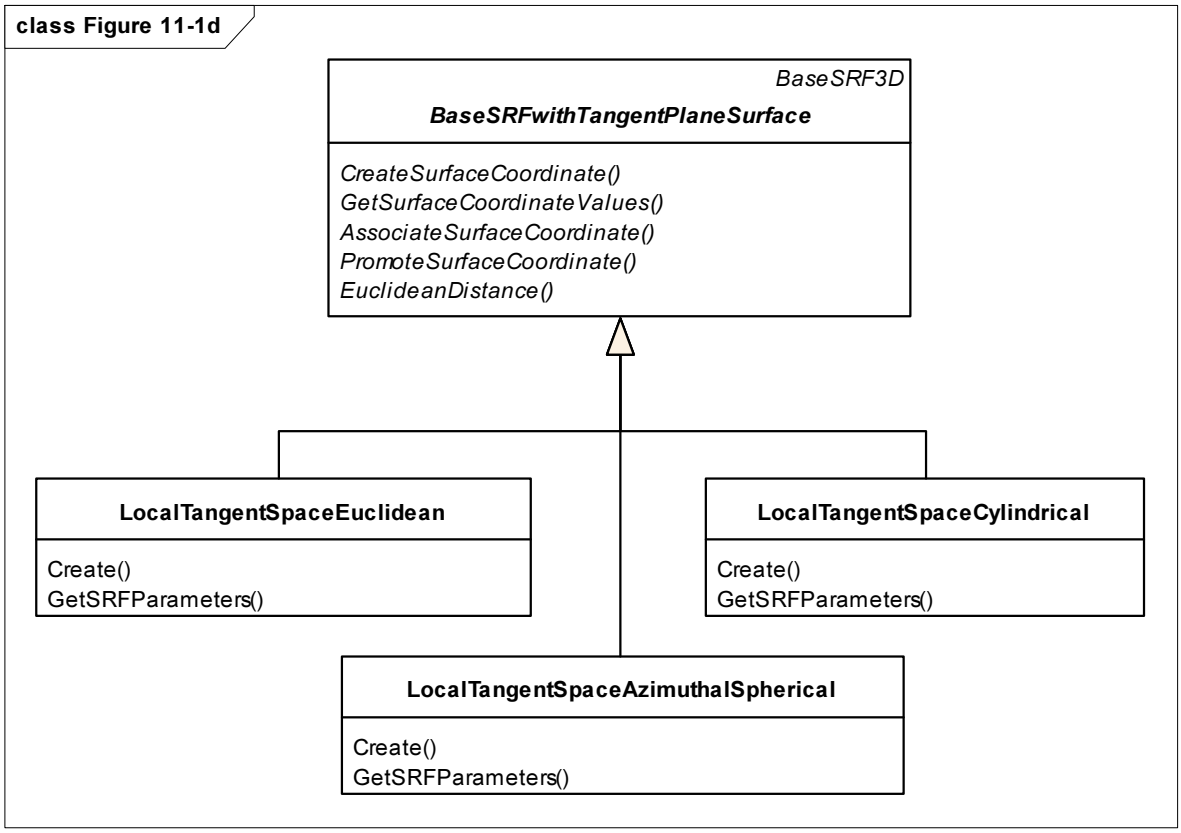

Figure 11.1 — Object inheritance hierarchy (continued)

11.8 Method precedence for life cycle objects and examples

There are restrictions on the order in which the methods of objects derived from `LifeCycleObject` may be invoked. The restrictions are:

a) The Create method (including subclassed versions) of a life cycle object shall be the first method invoked on any instance of the object.

b) The `Destroy` method (including subclassed versions) of a life cycle object shall be the last method invoked on any instance of the object. Depending on the language binding and the language capabilities, invocation of the destroy method may be:

   1) explicit – invoked by the API user,

   2) implicit – managed by the runtime system, or

   3) implicit/optionally explicit – managed by the runtime system if not explicitly invoked by the API user on any single instance.

c) All other methods shall only be invoked after the `Create` method and before the `Destroy` method.

The following examples illustrate the method sequence for life cycle objects.

NOTE The status for each invocation of a method or function should be checked for error conditions. For brevity and clarity this status checking is not shown in these examples.

EXAMPLE 1 Find the Euclidean distance between two locations.

```
--Note: Label in italics denotes a symbolic constant for this example –
Celestiodetic method Create(ORM_N_AM_1983, RT_N_AM_1983_CONUS; Output: ex1_srf)
ex1_srf method CreateCoordinate3D( Inputs: -77°(π/180°), +38°(π/180°), 0;
                                   Output: coordinate1)
ex1_srf method CreateCoordinate3D( Inputs +3°(π/180°), +49°(π/180°), 0;
                                   Output: coordinate2)
ex1_srf method EuclideanDistance( Inputs coordinate1, coordinate2;
                                  Output: distance)
-- use distance result --
coordinate1 method destroy
coordinate2 method destroy
ex1_srf method destroy
```

EXAMPLE 2 Change SRF representation of a location from UTM to Geocentric

```
--Note: Labels in italics denote symbolic constants for this example --
Function    CreateSRFSetMember
                ( Inputs: ( SRFS_UNIVERSAL_TRANSVERSE_MERCATOR,
                            SRFSM_ZONE_23_NORTHERN_HEMISPHERE ),
                          ORM_N_AM_1983,
                          RT_N_AM_1983_CONUS,
                  Output: source_srf)
source_srf method CreateCoordinate3D( Inputs: 350000, 400, 0,
                                      Output: source_coordinate)
Function CreateStandardSRF( Inputs: SRF_GEOCENTRIC_WGS_1984,
                                    RT_WGS_1984_IDENTITY,
                            Output: target_srf)
target_srf method ChangeCoordinate3DSRF( Inputs: source_srf, source_coordinate,
                                         Outputs: target_coordinate, region)
-- check region value and use result --
source_coordinate method destroy
target_coordinate method destroy
source_srf method destroy
target_srf method destroy
```

11.9 Data storage structures

11.9.1 Introduction

The data storage structures specify the exact ordering sequence and size of the information for persistent storage in any mass storage media. These structures are defined within this International Standard for applications that store SRM data and are not used with API methods or functions defined in this International Standard.

11.9.2 SRFT_Parameters

This non-object data type specifies the parameters for an arbitrary SRF template.


```
SRFT_Parameters ::= ( template_code     SRFT_Code )
{
  orm_code         ORM_Code;
   [
    CELESTIOCENTRIC:
      cc_srf_parameters               <empty>;
    LOCAL_SPACE_RECTANGULAR_3D:
      lsr_3d_srf_parameters           LSR_3D_Parameters;
    CELESTIODETIC:
       cd_srf_parameters              <empty>;
    PLANETODETIC:
      pd_srf_parameters               <empty>;
    LOCAL_TANGENT_SPACE_EUCLIDEAN:
      ltse_srf_parameters             LTSE_Parameters;
    LOCAL_TANGENT_SPACE_AZIMUTHAL_SPHERICAL:
      ltsas_srf_parameters            Local_Tangent_Parameters;
    LOCAL_TANGENT_SPACE_CYLINDRICAL:
      ltsc_srf_parameters             Local_Tangent_Parameters;
    LOCOCENTRIC_EUCLIDEAN_3D:
      lce_3d_srf_parameters           LCE_3D_Parameters;
    CELESTIOMAGNETIC:
      cm_srf_parameters               <empty>;
    EQUATORIAL_INERTIAL:
      ei_srf_parameters               <empty>;
    SOLAR_ECLIPTIC:
      sec_srf_parameters              <empty>;
    SOLAR_EQUATORIAL:
      seq_srf_parameters              <empty>;
    SOLAR_MAGNETIC_ECLIPTIC:
      sme_srf_parameters              <empty>;
    SOLAR_MAGNETIC_DIPOLE:
      smd_srf_parameters              <empty>;
    HELIOSPHERIC_ARIES_ECLIPTIC:
      haec_srf_parameters             <empty>;
    HELIOSPHERIC_EARTH_ECLIPTIC:
      heec_srf_parameters             <empty>;
    HELIOSPHERIC_EARTH_EQUATORIAL:
      heeq_srf_parameters             <empty>;
    MERCATOR:
      m_srf_parameters                M_Parameters;
    OBLIQUE_MERCATOR_SPHERICAL:
      oms_srf_parameters              Oblique_Mercator_Parameters;
    TRANSVERSE_MERCATOR:
      tm_srf_parameters               TM_Parameters;
    LAMBERT_CONFORMAL_CONIC:
      lcc_srf_parameters              LCC_Parameters;
    POLAR_STEREOGRAPHIC:
      ps_srf_parameters               PS_Parameters;
    EQUIDISTANT_CYLINDRICAL:
      ec_srf_parameters               EC_Parameters;
```

```
    LOCAL_SPACE_RECTANGULAR_2D:
      lsr_2d_srf_parameters          LSR_2D_Parameters;
    LOCAL_SPACE_AZIMUTHAL:
      lsa_srf_parameters             <empty>;
    LOCAL_SPACE_POLAR:
      lsp_srf_parameters             <empty>;
  ]
}
```

11.9.3 SRFS_Info

This non-object data type specifies the parameters for an arbitrary SRF set.

```
SRFS_Info ::= {
   orm_code                        ORM_Code;
   srfs_code_info                  SRFS_Code_Info;
}
```

11.9.4 SRF_Parameters_Info_Code

This non-object selection data type specifies an SRF code type.

```
SRF_Parameters_Info_Code ::= (  < 1 : // implementation dependent,
                                  1 : TEMPLATE,
                                  2 : SET,
                                  3 : INSTANCE,
                                > 3 : // reserved for registration )
```

11.9.5 SRF_Parameters_Info

This non-object data type specifies the parameters for an arbitrary SRF, SRF set or SRF template.

```
SRF_Parameters_Info ::= ( srf_parameters_info_code    SRF_Parameters_Info_Code )
{
  rt_code          RT_Code;
 [TEMPLATE:        srf_template          SRFT_Parameters;
  SET:             srf_set               SRFS_Info;
  INSTANCE:        srf_instance          SRF_Code;
 ]
}
```

11.9.6 SRF_Reference_Surface_Info

This non-object data type specifies the information for an arbitrary SRF with its associated DSS information.

```
SRF_Reference_Surface_Info ::= {
  dss_code                          DSS_Code;
  srf_parameters_info               SRF_Parameters_Info;
}
```

11.9.7 Coordinate structures

Structures are defined to store the specific values of a coordinate.

11.9.7.1 CD_3D_Coordinate

This non-object data type specifies the 3D coordinate-components for SRFT CELESTIODETIC.

```
CD_3D_Coordinate ::= {
   longitude                        Long_Float;
   latitude                         Long_Float;
   ellipsoidal_height               Long_Float;
}
```

11.9.7.2 CD_Surface_Coordinate

This non-object data type specifies the surface coordinate-components for SRFT CELESTIODETIC.

```
CD_Surface_Coordinate ::= {
   longitude                        Long_Float;
   latitude                         Long_Float;
}
```

11.9.7.3 EI_3D_Coordinate

This non-object data type specifies the 3D coordinate-components for SRFT EQUATORIAL_INERTIAL.

```
EI_3D_Coordinate ::= {
   right_ascension                  Long_Float;
   declination                      Long_Float;
   radius                           Long_Float;
}
```

11.9.7.4 Euclidean_2D_Coordinate

This non-object data type specifies the 2D coordinate-components for Euclidean space SRFTs.

```
Euclidean_2D_Coordinate ::= {
   u                                Long_Float;
   v                                Long_Float;
}
```

11.9.7.5 Euclidean_3D_Coordinate

This non-object data type specifies the 3D coordinate-components for Euclidean space SRFTs.

```
Euclidean_3D_Coordinate ::= {
   u                                Long_Float;
   v                                Long_Float;
   w                                Long_Float;
}
```

11.9.7.6 LSA_2D_Coordinate

This non-object data type specifies the 2D coordinate-components for SRFT LOCAL_SPACE_AZIMUTHAL_2D.

```
LSA_2D_Coordinate ::= {
   azimuth                          Long_Float;
   radius                           Long_Float;
}
```

11.9.7.7 LSP_2D_Coordinate

This non-object data type specifies the surface coordinate-components for SRFT LOCAL_SPACE_POLAR_2D.

```
LSP_2D_Coordinate ::= {
   radius                          Long_Float;
   angle                           Long_Float;
}
```

11.9.7.8 LTSAS_3D_Coordinate

This non-object data type specifies the 3D coordinate-components for SRFT LOCAL_TANGENT_SPACE_AZIMUTHAL_SPHERICAL.

```
LTSAS_3D_Coordinate ::= {
   azimuth                         Long_Float;
   angle                           Long_Float;
   radius                          Long_Float;
}
```

11.9.7.9 LTSAS_Surface_Coordinate

This non-object data type specifies the surface coordinate-components for SRFT LOCAL_TANGENT_SPACE_AZIMUTHAL_SPHERICAL.

```
LTSAS_Surface_Coordinate ::= {
   azimuth                         Long_Float;
   angle                           Long_Float;
}
```

11.9.7.10 LTSC_3D_Coordinate

This non-object data type specifies the 3D coordinate-components for SRFT LOCAL_TANGENT_SPACE_CYLINDRICAL.

```
LTSC_3D_Coordinate ::= {
   angle                           Long_Float;
   radius                          Long_Float;
   height                          Long_Float;
}
```

11.9.7.11 LTSC_Surface_Coordinate

This non-object data type specifies the surface coordinate-components for SRFT LOCAL_TANGENT_SPACE_CYLINDRICAL.

```
LTSC_Surface_Coordinate ::= {
   angle                           Long_Float;
   radius                          Long_Float;
}
```

11.9.7.12 LTSE_3D_Coordinate

This non-object data type specifies the 3D coordinate-components for SRFT LOCAL_TANGENT_SPACE_EUCLIDEAN.

```
LTSE_3D_Coordinate ::= {
   x                                Long_Float;
   y                                Long_Float;
   height                           Long_Float;
}
```

11.9.7.13 LTSE_Surface_Coordinate

This non-object data type specifies the surface coordinate-components for SRFT LOCAL_TANGENT_SPACE_EUCLIDEAN.

```
LTSE_Surface_Coordinate ::= {
   x                                Long_Float;
   y                                Long_Float;
}
```

11.9.7.14 Map_Projection_3D_Coordinate

This non-object data type specifies the 3D coordinate-components for map projection SRFTs.

```
Map_Projection_3D_Coordinate ::= {
   easting                          Long_Float;
   northing                         Long_Float;
   ellipsoidal_height               Long_Float;
}
```

11.9.7.15 Map_Projection_Surface_Coordinate

This non-object data type specifies the surface coordinate-components for map projection SRFTs.

```
Map_Projection_Surface_Coordinate ::= {
   easting                          Long_Float;
   northing                         Long_Float;
}
```

11.9.7.16 Equatorial_Spherical_3D_Coordinate

This non-object data type specifies the 3D coordinate-components for equatorial spherical SRFTs.

```
Equatorial_Spherical_3D_Coordinate ::= {
   longitude                        Long_Float;
   latitude                         Long_Float;
   radius                           Long_Float;
}
```

11.9.7.17 PD_3D_Coordinate

This non-object data type specifies the 3D coordinate-components for SRFT PLANETODETIC.

```
PD_3D_Coordinate ::= {
    latitude                        Long_Float;
    pd_longitude                    Long_Float;
    ellipsoidal_height              Long_Float;
}
```

11.9.7.18 PD_Surface_Coordinate

This non-object data type specifies the surface coordinate-components for SRFT PLANETODETIC.

```
PD_Surface_Coordinate ::= {
    latitude                        Long_Float;
    pd_longitude                    Long_Float;
}
```

11.9.8 Spatial_Coordinate_Code

This non-object selection data type specifies the spatial coordinate type.

```
Spatial_Coordinate_Code ::= (     < 0  :    // implementation dependent,
                                    0  : UNSPECIFIED,
                                    1  : CC_3D,
                                    2  : CD_3D,
                                    3  : CD_SURFACE,
                                    4  : CM_3D,
                                    5  : EC_AUGMENTED_3D,
                                    6  : EC_SURFACE,
                                    7  : EI_3D,
                                    8  : HAEC_3D,
                                    9  : HEEC_3D,
                                   10  : HEEQ_3D,
                                   11  : LCC_AUGMENTED_3D,
                                   12  : LCC_SURFACE,
                                   13  : LCE_3D,
                                   14  : LSA_2D,
                                   15  : LSP_2D,
                                   16  : LSR_2D,
                                   17  : LSR_3D,
                                   18  : LTSAS_3D,
                                   19  : LTSAS_SURFACE,
                                   20  : LTSC_3D,
                                   21  : LTSC_SURFACE,
                                   22  : LTSE_3D,
                                   23  : LTSE_SURFACE,
                                   24  : M_AUGMENTED_3D,
                                   25  : M_SURFACE,
                                   26  : OMS_AUGMENTED_3D,
                                   27  : OMS_SURFACE,
                                   28  : PD_3D,
                                   29  : PD_SURFACE,
                                   30  : PS_AUGMENTED_3D,
                                   31  : PS_SURFACE,
                                   32  : SEC_3D,
                                   33  : SEQ_3D,
                                   34  : SMD_3D,
                                   35  : SME_3D,
                                   36  : TM_AUGMENTED_3D,
                                   37  : TM_SURFACE,
                                  > 37   : // reserved for registration )
```

11.9.9 Coordinate

This non-object data type stores one of the defined coordinate.

```
Coordinate ::=  (   spatial_coord_code  Spatial_Coordinate_Code )
{
 [ CC_3D:                   cc_3d           Euclidean_3D_Coordinate;
   CD_3D:                   cd_3d           CD_3D_Coordinate;
   CD_SURFACE:              cd_surface      CD_Surface_Coordinate;
   CM_3D:                   cm_3d           Equatorial_Spherical_3D_Coordinate;
   EC_AUGMENTED_3D:         ec_aug_3d       Map_Projection_3D_Coordinate;
   EC_SURFACE:              ec_surface      Map_Projection_Surface_Coordinate;
   EI_3D:                   ei_3d           Equatorial_Inertial_3D_Coordinate;
   HAEC_3D:                 haec_3d         Equatorial_Spherical_3D_Coordinate;
   HEEC_3D:                 heec_3d         Equatorial_Spherical_3D_Coordinate;
   HEEQ_3D                  heeq_3d         Equatorial_Spherical_3D_Coordinate;
   LCC_AUGMENTED_3D         lcc_aug_3d      Map_Projection_3D_Coordinate;
   LCC_SURFACE:             lcc_surface     Map_Projection_Surface_Coordinate;
   LCE_3D:                  lce_3d          Euclidean_3D_Coordinate;
   LSA_2D:                  lsa_2d          LSA_2D_Coordinate;
   LSP_2D:                  lsp_2d          LSP_2D_Coordinate;
   LSR_2D:                  lsr_2d          Euclidean_2D_Coordinate;
   LSR_3D:                  lsr_3d          Euclidean_3D_Coordinate;
   LTSAS_3D:                ltsas_3d        LTSAS_3D_Coordinate;
   LTSAS_SURFACE:           ltsas_surface   LTSAS_Surface_Coordinate;
   LTSC_3D:                 ltsc_3d         LTSC_3D_Coordinate;
   LTSC_SURFACE:            ltsc_surface    LTSC_Surface_Coordinate;
   LTSE_3D:                 ltse_3d         LTSE_3D_Coordinate;
   LTSE_SURFACE:            ltse_surface    LTSE_Surface_Coordinate;
   MERCATOR_AUGMENTED_3D:   m_aug_3d        Map_Projection_3D_Coordinate;
   MERCATOR_SURFACE:        m_surface       Map_Projection_Surface_Coordinate;
   OMS_AUGMENTED_3D:        oms_aug_3d      Map_Projection_3D_Coordinate;
   OMS_SURFACE:             oms_surface     Map_Projection_Surface_Coordinate;
   PD_3D:                   pd_3d           PD_3D_Coordinate;
   PD_SURFACE:              pd_surface      PD_Surface_Coordinate;
   PS_AUGMENTED_3D:         ps_aug_3d       Map_Projection_3D_Coordinate;
   PS_SURFACE:              ps_surface      Map_Projection_Surface_Coordinate;
   SEC_3D:                  sec_3d          Equatorial_Spherical_3D_Coordinate;
   SEQ_3D:                  seq_3d          Equatorial_Spherical_3D_Coordinate;
   SMD_3D:                  smd_3d          Euclidean_3D_Coordinate;
   SME_3D:                  sme_3d          Euclidean_3D_Coordinate;
   TM_AUGMENTED_3D:         tm_aug_3d       Map_Projection_3D_Coordinate;
   TM_SURFACE:              tm_surface      Map_Projection_Surface_Coordinate;
 ]
}
```

11.9.10 RD_Code

`RD_Code` is a `Selection` data type (see 11.2.7). `RD_Code` specifies the RD code associated with a specified RD as defined in Clause 7. Table 7.3 is a directory of RD specifications, each of which includes a code value and a corresponding label. In the API, a standard or registered RD is represented by its RD code.

11.9.11 OBRS_Code

`OBRS_Code` is a `Selection` data type (see 11.2.7). `OBRS_Code` specifies an OBRS code defined in Clause 7. Table 7.36 is a directory of OBRS specifications, each of which includes a code value and a corresponding label. An OBRS is an SRM concept that is not directly used in the functional API.

http://standards.iso.org/ittf/PubliclyAvailableStandards/