

Our ref.: JA

Date: 14 November 2003

Secretariat of ISO/IEC JTC 1/SC 24

Computer Graphics and Image Processing

Title:	Collation of comments ISO/IEC CD 18026
Source:	ISO/IEC JTC 1/SC 24 Secretariat
Replaces:	--
Document Type:	Comments Collation
Projects:	ISO/IEC CD 18026
Status:	CD comments from internal enquiry.
Action ID:	For information and action by WG.
Due Date:	--
Distribution:	P, O, L, Members Secretariat JTC 1, ITTF, SC 24 Chairman and WG Convenors
Date of Distribution:	14 Nov 2003
Distribution Medium:	Web

ISO/IEC CD 18026 list of comments**USA Comments****GENERAL****US_G001:**

Problem: Most of the links in the Table of Contents do not work.

Recommendation: Need to be fixed.

US_G002:

All Clauses:

Problem: There are many instances in the International Standard where appropriate commas are missing. Many have been identified in the CD comments. Others may still be undetected.

Recommendation: Editors should pay closer attention to the use of commas for consistency, clarity, and readability

US_G003:

Problem: Both the articles ‘a’ and ‘an’ are used with RD. Need to be consistent.

Recommendation: Use the correct article in accordance with correct English usage.

US_G004:**Annex J:**

Problem: Based on action item to identify retired versus active datums for prioritizing a small list of support datums, NIMA G&G recommends do not attempt to include all datums in the 18026 standard, but instead recommend the general use of WGS84.

Recommendation: Remove Annex J and move this information to appropriate Registries.

Rationale: The G&GWG feels that they cannot support the action on Laura Moore and Paul Foley to provide the classification of retired versus active datums because the NIMA cannot arbitrarily designate the active v.s. superceded status for the complete list; this is a native use issue for each nation. (Notes from June 12 G&GWG and submitted against WG7 of SRM.)

US_G005:**Annex J:**

Problem: NIMA cannot provide deprecation information for ERMs outside the United States. This information should be requested from the appropriate agency(ies) within the countries where a particular ERM is/has been used.

Recommendation: The NIMA G&GWG recommends including ORM_WGS_1960 and ORM_WGS_1966 in the list of deprecated ORMs.

Rationale: WGS 1960 and WGS 1966 are no longer used.

US_G006**Figures**

Problem: Readability of figures when colour is used (example Figure 5.6)

Recommendation: Figures currently have colouring. Patterns should be used where possible for readability when printed in black & white. Review all figures for colouring and correct where needed.

TECHNICAL

Clause 2: Normative References

US_T001:

Problem/recommendation: include the ISO registry that is used by 18026 in the Normative References

Rationale: completeness. Actually a reference is included for ISO/IEC 9973 in Clause 2.

If we add a general reference to the 19100 series of ISO standards, then ISO 19135 would also be included which covers registries related to geographic information.

Clause 3 Terms, definitions, symbols, and abbreviated terms

US_T002:

Table 3.2

Problem: Should not there be a spherical longitude or just longitude also or is the same symbol used for both?

Recommendation: There should be only one longitude and no celestiodetic longitude. Remove celestiodetic from the definition.

US_T003:

Clause 3, Table 3.3: EGM

Problem/recommendation: EGM may also represent Earth Gravitational Model. Change to read as "Gravitational" and add a definition for Earth Gravitational Model. Supporting: EGM96 is a spherical harmonic expansion of the disturbing potential of the gravitational field; rotational effects are not included in this model. Gravity includes rotational effects.

Rationale: Correctness

US_T004:

Clause 3, Section 3.3.6: geodetic azimuth

Problem/recommendation: Clause should include instructional text about conventions for measuring azimuths clockwise or counterclockwise and from the north or south side of the meridian.

Rationale: Completeness and accuracy of use

Clause 4 Concepts

US_T005:

Clause 4, Section 4.1.g-h

Problem/recommendation: Add Text The distinction between a **transformation** involves a change in reference datum versus a **conversion** that only involves a change in coordinate geometry. Changing reference datums is an empirical process based on physical measurements and always involves some level of error. Conversions are analytical functions that can be performed with any degree of accuracy and fidelity necessary, albeit occasionally at the cost of speed and efficiency. Note: there is a related comment for Clause 10.

Rationale: The standard may benefit from making this distinction, which is identified in ISO 19111. Provides clarity.

US_T006:

Clause 4, Section 4.2 and Figure 4.1 (and also in Clause 8):

Problem: Celestial objects are defined as being "of such a scale as to be of interest to astronomers." This is a rather loose definition. Using this definition, one could probably classify an Earth based telescope as a celestial object. The smallest celestial object tracked by astronomers is 5 – 10 meters across, <http://www.oarval.org/closest.htm>. Also the definition should refer to something like heavenly space, outer space, or similar term. There also should be no dependence on scale, but there should be some statement as

to not being man made or being a "natural: object. Do not think that man-made objects in orbit are considered celestial objects. Recommend dropping the distinction between celestial and non-celestial objects. This information can be captured in the properties of the object, e.g., object X is a satellite of object Y.

Recommendation: There is no benefit in complicating matters by distinguishing between celestial and non-celestial objects. A planet is a satellite of a star. The ISS is a satellite of the Earth. The space shuttle is a satellite while in orbit and a non-celestial object when it is not in orbit. Stars are typically satellites of a galactic core. Making these arbitrary distinctions is not worthwhile. Any physical or conceptual object, regardless of size, shape, type of motion, locomotive capability, or any other physical or abstract property can be used in the definition of an object space. At a minimum, Suggested rewording –

"Celestial objects are ~~of such a scale as to be of interest to astronomers~~ natural objects that are located external to the Earth's atmosphere."

Also, the US NB understands other comments were generated on this subject. The US NB believes the WG needs to consider those at the same time because the other comment may provide a superior solution.

US_T007:

Figure 4.1

Problem: It is not clear if "satellite" refers to "natural" satellites like the moon or does it also include artificial satellite such as a GOES satellite?

Recommendation: Make meaning of satellite clear.

US_T008:

4.5 Object reference models, Example 2, 2nd item a

Problem: The constraint is poorly worded.

Recommendation: Believe that the correct wording should be:

"The constructed directed line bound to the z-axis reference datum ~~that~~ contains the centre of the constructed oblate spheroid that is bound to the oblate spheroid reference datum."

US_T009:

Clause 4, Section 4.5 Example 6 and Section 4.7 Example 3 has the same issue:

Problem: They indicate that European Datum 1950 Mean Solution is a realization of the template in 4.5 Example 2 using the International 1924 oblate spheroid reference datum. There is only one European Datum 1950 realization. Mean Solution refers to a specific estimate of the binding parameters between European Datum 1950 and the World Geodetic System 1984. This example incorrectly avoids the distinction between a geodetic datum (object reference model for the Earth) and the datum transformation (binding) between geodetic datums (object reference models). These are separate and distinct concepts.

Recommendation: Delete "Mean Solution" following European Datum 1950 and change "ORM_EUR_1950_MEAN_SOLUTION" to "ORM_EUR_1950". See comments regarding Annex E for related changes.

Rationale: Completeness of information provided in example

US_T010:

4.6.1 Abstract coordinate systems, Example 3

Problem: No reference to Figure 4.9

Recommendation: Add the following sentence between the 1st and 2nd sentences.

"This coordinate system is illustrated in Figure 4.9."

US_T011:

4.6.3 Spatial coordinate systems

Problem: Is the embedding normal or orthonormal? The text says normal and the Figure 4.10 says orthonormal. Also there should be a reference to Figure 4.10 in the text.

Recommendation: Added reference to figure and be consistent.

Clause 5 Coordinate systems

US_T012:

5.1.1 Introduction

Problem: This paragraph is not clear. This clause is titled "Coordinate systems" but refers to Clause 8 for spatial coordinate systems and implying that this clause covers abstract coordinate systems. In Clause 4.1, item a, it refers to Clause 5 for spatial coordinate systems. Clause 8 describes spatial reference frames.

Recommendation: Rewrite to make it clear what is covered in this clause.

US_T013:

5.2.2 Abstract CS, 2nd item c, last 2 sentences

Problem: The sentence structure of the two sentences is incorrect. In the penultimate sentence it says "...range shall be a subset a smooth surface." And the last sentence says "...range shall be a subset an implicitly specified smooth curve." Both can be made to read better by the insertion of the preposition "of". However, it is not clear this is what is meant.

Recommendation: Correct sentence structure to make meaning clear.

US_T014:

Clause 5.3.6.1:

Problem/recommendation: References to ellipsoidal heights are not consistent with the discussions in Clause 9 related to vertical offset surface.

Rationale: Replace references to "ellipsoidal heights" with "spheroidal heights" to maintain consistency with clause 9, section 9.2.4.

US_T015:

Table 5.1

Problem: Should not the footnote on "curve" also be on "plane curve" since the footnote refers to both curve and plane curve?

Recommendation: Fix as appropriate.

Clause 6 Temporal Coordinate Systems

US_T016:

Clause 6.2.3:

Problem: Text needs to be included that address real-time usage of UTC.

Recommendation: Insert the text at the end of the paragraph: "Due to the latency of the UTC(BIPM) values, one of these timing centers must be referenced for real-time realization of UTC."

Rationale: Completeness. UTC, as determined by BIPM is not published for a period of time after the epoch in question. For real-time use of UTC, it must be realized at a specific center.

Clause 9

US_T017:

Clause 9, Section 9.2:

Problem: References to ellipsoidal heights are not consistent with the discussions in Clause 9 related to vertical offset surface.

Recommendation: Replace references to "ellipsoidal heights" with "spheroidal heights" to maintain consistency with clause 9, section.

Rationale: Consistency.

US_T018:

Clause 9, Section 9.2.3: paragraph 3, sentence 2

Problem/recommendation: Replace “A *geoid* is a model of the Earth’s gravity equipotential surface at mean sea level associated to an ERM.” with “The *geoid* is a specific equipotential surface of the Earth’s gravity field that best fits the global mean sea surface in a minimum variance sense. The geoid cannot be measured directly. Global, regional, and local approximations of the geoid are developed from empirical measurements in association with specific ERMs.”

Rationale: Correctness

US_T019:

Clause 9, Section 9.4:

Problem: In the note following Figure 9.4 it states that SRTM does not use the WGS 84 ellipsoid. NIMA specified the use of the EGM96 geoid for the processing of the SRTM data. EGM96 is defined relative to the WGS 84 Ellipsoid so I believe this statement is incorrect. Also, many GPS receivers include a crude model of the WGS 84 geoid based on a $10^\circ \times 10^\circ$ of geoid heights. Therefore, depending on the users setting in their receiver, the height may be referenced to the WGS 84 ellipsoid or to an approximation of the WGS 84 (EGM84) Geoid.

Recommendation: N/A

Rationale: correctness

Clause 10

US_T020:

Clause 10, Paragraph 10.3.1:

Problem: It is very important to understand that coordinates for real world objects associated with a particular ORM are subject to varying degrees of accuracy based on a wide range of factors associated with the physical measurement processes associated with determining coordinates. Measurements are typically made relative to a network of control points. The positions of these control points are determine at the time the ORM is defined based on the most rigorous measurement methods available at the time. The resulting network of control points includes various types of distortions. These distortions propagate into coordinates determined through measurements relative to the control points. The affine transformation documented in this paragraph assumes that the coordinates referenced to a particular ORM are free from error. A caveat should be added to make this clear.

Recommendation: Insert the following sentence after the third sentence: “The processes by which ORMs for the Earth are established are all based on physical measurements. These measurements are subject to error and therefore introduce various types of distortions into the ORM. Equation 10.1 assumes coordinates in ORM_s are error free and does not compensate for these distortions.”

Rationale: Completeness

Clause 12

US_T021:

Clause 12.2.3:

Problem: standard states “Codes for registered items must be numerical”.

Recommendation: Please clarify in the 18026 standard *which* codes for registered items must be numerical.

Rationale: The ISO standard for registration is not included in the Normative References. NIMA G&GWG is concerned to ensure that the numerical codes for registered items are the unique identifiers that are essentially invisible to the user and are not intended to replace the standardized and agreed upon codes used in the Geodesy & Geophysics community; NIMA has many codes, including datum and ellipsoids that are not numerical.

Annex E

US_T022:

Annex E, Table E.4:

Problem: An Earth Reference Models (ERM) is an Object Reference Model (ORM) for the Earth. It would be appropriate in this table to describe the binding constraints for each ERM. The binding of each ERM to ERM_R (ORM_WGS_1984) should be:

- Stored in an appropriate Registry of this information, or
- Listed in a separate table.

NOTE: There is only one binding between the Earth and any given ERM, whereas, many estimates may exist for the binding between any two ERMs. The later are constantly subject to revision based on the acquisition of new information.

Recommendation: Split Table E.4 into two tables. The first table should contain the binding constraints for each ERM (i.e., origin point coordinates, reference azimuths, origin deflections of the vertical, origin geoid heights, etc.). The second table should contain the bindings of these ERMs to the reference ERM (ORM_WGS_1984).

The NIMA G&GWG recommends NOT including the second table, as these binding parameters are not unique and are subject to change. The NIMA G&GWG recommends storing this information in an appropriate Registry.

If bindings to ERM_R are listed, then the following information must be included for each set of binding parameters:

accuracy estimate for each parameter
transformation model
date of computation
intended use
number of control points used to determine binding
citation for the source of the binding parameters

The accuracy estimates are based on errors in the coordinates used to determine the binding and the distortions for which the transformation model cannot compensate.

The NIMA G&GWG recommends that if the bindings to ERM_R are listed, only the available parameters should be given. In other words, if only the origin shifts ΔX , ΔY , and ΔZ are known, only these parameters should be listed in the table.

Rationale: Each geodetic datum is a binding of a coordinate system to all or part of the Earth. This binding is defined at the time the geodetic datum is determined. There are many techniques used to perform this binding. Some require the definition or selection of an oblate spheroid (i.e., ellipsoid) and some do not. At some later date, it may be necessary to relate one geodetic datum (ORM) to another. A binding can be determined between the two ORMs provided that a sufficient number of coordinates for common physical locations on the object Earth are known with respect to both ORMs. While the bindings between the Earth and each unique ORM are uniquely defined at the time the ORM is established, the bindings between ORMs are not unique. Many bindings between different ORMs are possible. Different sets of coordinates, different measurement methods, different computational methods, etc. all contribute to the large number of possible bindings between any two ORMs. This standard does not properly take this distinction into account, generally treating bindings between ORMs as unique ORMs.

Inclusion of empirical data, which is subject to change as new information become available, into a standard implies that values are constants. This can lead to “hard-wiring” of values into systems developed in compliance with the standard. Placing this information in Registries implies that it is subject to change and system should be designed to use the latest information available from official Registries.

Empirical data includes inherent errors. The accuracy of these parameters must be included to allow users to determine if the data meets their requirements.

Additional notes from comment provider: The 3-parameter transformations given NIMA TR 8350.2 are computed without any attempt to compute rotations and scale differences. In reality, there are probably rotation and scale differences between the local datums and WGS 84. These parameters are listed as zero only because NIMA did not attempt to estimate them. If all the parameters had been computed, different values for the translation parameters would have been determined. It is critical to understand the parameters given in TR 8350.2 DO NOT model all of the differences between the local datums and WGS 84. The difference is modeled well enough to produce maps and charts at 1:250K scale or smaller. Adding accuracy

information for binding parameters allows users to make their own judgment regarding the appropriate applications. A statement like, “Binding parameters listing 83502T as a reference were developed for mapping and charting applications and are not suitable for high-accuracy applications.” should be added.

US_T023:**Annex E:**

Problem: The binding parameters between ORM_MIDWAY_ASTRO_1961 and WGS 84 are in error.

Recommendation: “ $\Delta x = 912$, $\Delta y = -58$, $\Delta z = 1227$, $\omega_1 = \omega_2 = \omega_3 = 0''$, $\Delta s = 0$.” to “ $\Delta x = 403$, $\Delta y = -81$, $\Delta z = 277$, $\omega_1 = \omega_2 = \omega_3 = 0''$, $\Delta s = 0$.”

Rationale: With assistance from the National Geodetic Survey, NIMA discovered an error in the Midway Astro 1961 coordinates for the single control station used to determine this binding. A new binding was computed. Note the magnitude of the error. This is an example of why NIMA strongly opposes the inclusion of bindings between ORMs in ISO 18026.

US_T024:**Annex E:** Table E.4

Problem/recommendation: **Date published** entries for all ORMs that do not include a year as part of their label should be changed to “TBD” pending more research.

Rationale: **Data published** field is sometimes populated with the publication date for the most recently determined binding to the standard ORM for the Earth (ORM_WGS_1984). This misleads users into believing that these ORMs were defined much more recently than they actually were. For example, ORM_ADINDAN_BURKINA_FASO is related to the Adindan or Blue Nile Datum of 1958. The binding to WGS 84 as determined by NIMA was published in 1991; however, the geodetic datum (ORM) appears to have been defined in 1958.

Annex J**US_T025:****Annex J:**

Problem: North American Datum of 1927 should not be deprecated. It is still in active by the U.S. Geological Survey for large sections of the United States.

Recommendation: Move entries for ORM_N_AM_1927* in Table J.7 to the appropriate table(s) in Annex E.

Rationale: NAD 1927 is still in use.

US_T026:**Annex J:**

Problem: The NIMA G&GWG strongly discourages the inclusion of empirical parameters in a standards document.

Recommendation:

Remove **Binding** information from Tables J.7, E.8, E.9, E.10, E.11, E.12, and E.13. Move this information to a Registry.

NOTE: If the committee determines that empirical data will not be removed, then the US NB strongly recommends inclusion of error estimates for all parameters (semi-axes, flattening, binding parameters, etc.).

Rationale:

Inclusion of empirical data, which is subject to change as new information become available, into a standard implies that values are constants. This can lead to “hard-wiring” of values into systems developed in compliance with the standard. Placing this information in Registries implies that it is subject to change and system should be designed to use the latest information available from official Registries.

Empirical data includes inherent errors. The accuracy of these parameters must be included to allow users to determine if the data meets their requirements.

EDITORIAL

Scope

US_E001:

Scope: Third paragraph, first sentence.

Problem: The number twenty is used incorrectly in the body of the sentence.

Recommendation: The application program interface supports more than ~~twenty~~ 20 forms of position representation.

Rationale: Consistency within document.

Clause 2 Normative references

US_E002:

Table

“*Conférence Générale des Poids et ~~Mesure~~ Mesures*”

Rationale: Copy error.

US_E003:

Change: Insert after I9973 reference, “I19100, ISO 19100 Series, *Geographic information*”

Rationale: Completeness

Clause 3 Terms, definitions, symbols, and abbreviated terms

US_E004:

Section 3.2: Correct subsection numbering

Change: Change numbers for paragraphs 3.3.1-3.3.9 to 3.2.x

Rationale: Proper formatting

US_E005:

Section 3.2: This section should be expanded to include additional terms commonly used throughout this standard.

Change: Add definitions for the terms below and renumber existing paragraphs as appropriate.

coordinate system (CS)

domain

Earth reference model (ERM)

embedding

object reference model (ORM)

object-space

the real or abstract universe that contains a spatial object

position-space

a vector space abstraction of object-space

range

reference datum (RD)

spatial operations

spatial reference frame

vertical offset surface

Rationale: Completeness; although definitions are provided for many terms as they are introduced, including brief definitions in this clause would provide a valuable reference. Hyperlinks to the appropriate sections in this standard would further improve the utility of Clause 3.

US_E006:

Table 3.3: UPS

Change: Capitalize “Earth” unless there is ISO guidance to the contrary. My reference: *The Chicago Manual of Style*, 14th edition, 1993.

Rationale: Correctness

US_E007:

Clause 3, Table 3.3: UTM

Change: Capitalize “Earth” unless there is ISO guidance to the contrary.

Rationale: Correctness

US_E008:

Table 3.3: Abbreviation FRG for Federal Republic of Germany is a cold war term.

Change: FRG

Rationale: Germany is common usage to describe country so FRG may be inappropriate.

Clause 4 Concepts

US_E009:

Paragraph 4.1.c: Why the parenthetical “(models of)”? Is this meant to indicate an “or” condition? Equivalent to “...to bind spatial coordinate systems to real world objects and/or models thereof.”

Change: Replace “...spatial coordinate systems to (models of) real world objects...” with “...spatial coordinate systems to real or conceptual objects...”

Rationale: Clarity

US_E010:

4.2, 1st paragraph, 2nd sentence.

Problem: Add a hyphen between “real” and “world”. There may be other instances of this.

Recommendation: The editors should review that each use is appropriate.

Rationale: The use of “real” in this instance is unnecessary and redundant.

US_E011:

4.3 Position space and normal embeddings, 1st para, 1st sentence

Problem: Missing article in sentence.

Recommendation: Change as shown.

"N-dimensional position-space is the n-dimensional Euclidean space."

US_E012:

Clause 4.4, second paragraph, second sentence (below Figure 4.3):

Problem: Awkward construction.

Change: A reference datum is bound when the reference datum is identified with a corresponding constructed entity in object-space. ~~Corresponding~~ This means that each reference datum is bound to a constructed entity of the same geometric object type; that is, points are bound to identified points, lines to constructed lines, curves to constructed curves, planes to constructed planes, and surfaces to constructed surfaces.

Rationale: Clarity.

Alternative: “Corresponding” means that each reference datum is bound to a constructed entity of the same geometric object type; that is, points are bound to identified points, lines to constructed lines, curves to constructed curves, planes to constructed planes, and surfaces to constructed surfaces.

US_E013:

Clause 4.5, Example 1, fourth sentence and fifth sentences:

Problem: Commas missing after prepositional phrases.

Recommendation: If the point lies in the directed line, then there is exactly one compatible embedding in 1D position-space. In 2D and 3D position-space, there will be infinitely many compatible embeddings if the point lies on the directed line.

Rationale: Consistency and correct sentence punctuation.

US_E014:

4.5 Object reference models, Example 2, 2nd item b

Problem: Missing verb.

Recommendation: Believe that the correct wording should be:

The constructed plane bound to the xz -plane reference datum contains the constructed directed line bound to the z -axis reference datum.

US_E015:

4.5, Example 3, paragraph before figure, 2nd sentence

Problem/Recommendation: “The equatorial plane of the oblate spheroid determines the xy -plane, and its intersection with the oblate spheroid axis of rotation determines the origin point and the z -axis. ”

Rationale: Subject-verb agreement.

US_E016:

Clause 4.6, Example 3, third sentence (below Figure 4.9):

Problem: Comma missing after introductory phrase.

Recommendation: When inverse mapping equations are combined with the surface geodetic coordinate system function, the result is also a surface coordinate system function.

Rationale: Consistency.

Clause 5 Coordinate systems

US_E017:

5.1.1 Introduction, 4th sentence

Problem: Missing article.

Recommendation: Modify as indicated.

"...into an object space ..."

US_E018:

5.2.2, 1st sentence

Problem/Recommendation: “a means of identifying a set of positions”

Rationale: Missing word.

US_E019:

5.2.2, 2nd b

Problem/Recommendation: “preserving function from the”

Rationale: Remove extra space between “from” and “the”.

US_E020:

5.2.2, 1st and 2nd footnotes, 2nd sentence

Problem/Recommendation: “for all points in the CS domain”

Rationale: Number agreement.

US_E021:

Clause 5, Section 5.2.2: Footnotes 1 and 2,

Problem: incorrect case used in second sentence.

Recommendation: Change "...smooth curve for all point in the..." to "smooth curve for all points in the..." in the second sentences of both footnotes.

Rationale: Correctness

US_E022:

5.2.2, 5th paragraph after the notes

Problem/Recommendation: "*CS parameters .*"

Rationale: Remove extra space between "parameters" and the period.

US_E023:

5.2.2, notes and footnotes

Problem/Recommendation: Indicate "Note 2". (Probably need to change Note 3 to Note 2.)

Rationale: The numbering is messed up. There is a "Note 1" and "Note 3" but no "Note 2". The 3 footnotes seem fine.

US_E024:

5.2.2, note 3

Problem/Recommendation: "rather than by geometric construction.)"

Rationale: There is no left paren so right paren not needed.

US_E025:

5.2.4.2, Example 1

Problem/Recommendation: "is the surface of the oblate spheroid"

Rationale: Missing word.

US_E026:

5.2.4.2 Coordinate surfaces and induced surface CSs, Example 2, 2nd sentence

Problem: Subject verb agreement.

Recommendation: Change as indicated.

"...~~is~~ are identical..."

US_E027:

Clause 5, Section 5.2.4.2:

Problem: Table reference in example 1 is incorrect.

Recommendation: Change "...Table 5.13..." to "...Table 5.14..."

Rationale: Correctness

US_E028:

Clause 5, Section 5.2.4.2:

Problem: Table reference in example 2 is incorrect.

Recommendation: Change "...Table 5.16..." to "...Table 5.17..."

Rationale: Correctness

US_E029:

5.2.5.1, footnote 5

Problem/Recommendation: "defines meridian as the intersection between"

Rationale: Missing word.

US_E030:

5.2.6, 1st paragraph

Problem/Recommendation: "In some applications of a CS in the context of a spatial reference frame (see Clause 8), it is necessary to consider a displaced and/or rotated version of a CS. To consider such a CS as

the CS of a spatial reference frame, a generating function for the displaced and rotated version of the CS must be specified. These modified versions of CS generating functions can be specified in a uniform way through the process of localization defined below.”

Rationale: Just missing 2 commas and incorrect verb tense.

US_E031:

5.3.3.1, 1st sentence

Problem/Recommendation: “has no direct significance with ~~the~~ respect to the geometry of position-space. (For example, the Euclidean distance between two surface geodetic coordinate tuples has no obvious meaning in position space.):-”

Rationale: Remove extra word. The phrase in parentheses is a complete sentence.

US_E032:

5.3.4, 1st paragraph, last sentence

Problem/Recommendation: “The second step is to associate the surface of the range to 2D coordinate-space”

Rationale: Missing word. See similar sentence 4.

US_E033:

5.3.4, 2nd paragraph, 1st sentence

Problem/Recommendation: “In the case of planar projection functions,”

Rationale: Missing word. See similar sentence next paragraph.

US_E034:

5.3.4, 3rd paragraph

Problem/Recommendation: ~~Homeomorphism~~ homeomorphism

Rationale: Misspelling.

US_E035:

Tables 5.10 through 5.33

Problem: The order in which the "Coordinates" are listed is frequently different from the order in which they are listed in the "Domain of the generating function or mapping equations" entry. For example, see Table e 5.10. This can lead to confusion because users will assume that they are listed in the same order and not look at them closely.

Recommendation: List them in the same order in both places and this will reduce the chances of error by the reader.

US_E036:

Tables 5.8 through 5.33

Problem: Sometimes the entries for the coordinate "Domain of the generating function or mapping equations" are listed on separate lines as in Table 5.10, and sometimes on a single line as in Table 5.15.

Recommendation: Should be the same through for consistency.

US_E037:

Table 5.10

Problem: Reference entry is not linked.

US_E038:

Table 5.28 and Table 5.29

Problem: The figures are not sharp.

Recommendation: Replace with better figures.

US_E039:

Clause 5, Table 5.28 (Field: Inverse of the generating function or mapping equations):

Problem: Comma missing in introductory phrase in first sentence under **Specification**.

Recommendation: For ϕ functional, iteration is used for the functional representation of the inverse map projection.

Rationale: Clarity.

Clause 6 Temporal coordinate systems

US_E040:

6.2.1, last paragraph, 1st sentence

Problem/Recommendation: “*coordinate system* is is a”

Rationale: Remove duplicate word.

US_E041:

6.2.1, Examples

Problem/Recommendation: The word “EXAMPLE” is capitalized in 1 and 3 and not in 2.

Rationale: Make the word “EXAMPLE” capitalized in 2 per directives.

US_E042:

6.2.2, 1st sentence

Problem/Recommendation: “of a set of ~~dynamical~~ dynamic temporal coordinate systems”

Rationale: Dynamical is rarely used in speech. Readers may think it is an error.

Clause 7

US_E043: All tables

Problem: The tables are inconsistent in their use of double and single lines of cell borders (in print view).

Recommendation: Make consistent.

US_E044:

Table 7.1, Directed Curve

Problem/Recommendation: The subscript after “t”. The rendering does not look to be consistent.

Rationale: The subscript after “t” does not seem to be rendered properly.

US_E045:

Clause 7.1, third paragraph, first sentence:

Problem: Comma missing in introductory phrase.

Recommendation: When reference datums are bound with properly constrained geometric constructs in object-space, a unique normal embedding is determined.

Rationale: Clarity.

US_E046:

7.2.3, 2nd sentence

Problem/Recommendation: “geometrically-defined” Remove italic.

Rationale: Font error.

US_E047:

Equation 7.2

Problem/Recommendation: “definition”

Rationale: Typo.

US_E048:

Clause 7.2.5, first sentence:

Problem: Awkward construction.

Recommendation: An RD is *bound* when the RD is identified with a corresponding constructed entity in object-space. ~~Corresponding~~ This means that each RD is bound to a constructed entity of the same geometric object type; that is, points are bound to identified points, directed lines to constructed lines, directed curves to constructed curves, oriented planes to constructed planes, and oriented surfaces to constructed surfaces.

Alternative: “Corresponding” means that each RD is bound to a constructed entity of the same geometric object type; that is, points are bound to identified points, directed lines to constructed lines, directed curves to constructed curves, oriented planes to constructed planes, and oriented surfaces to constructed surfaces.

Rationale: Clarity.

US_E049:

Section 7.3.3 10.3.1:

Problem: Cross-reference to non-existent portion of document

Recommendation: Delete NOTE 2 following Figure 7.5

Rationale: Correctness

US_E050:

7.4.2, 4th paragraph, end

Problem/Recommendation: “the ORM is a *global model*.”

Rationale: Missing sentence period.

US_E051:

Table 7.8

Problem: There are two header rows for the table.

Recommendation: Remove one of them.

US_E052:

Table 7.9, ORMT_3D_PROLATE_-SPHEROID, BC3

Problem/Recommendation: Remove italics from “BC3” and correct spelling of 2nd “metres”.

Rationale: Typos.

US_E053:

Table 7.9, ORMT_3D_BI_-AXIS_ORIGIN, BC2

Problem/Recommendation: “The constructed directed lines bound ~~the~~ to RD 2 and RD 3 shall be perpendicular.”

Rationale: Typo.

US_E054:

Table 7.10

Problem: The definition for Region is split across the page brake.

Recommendation: Ensure that table cells are not split across page breaks.

US_E055:

Table 7.10, 4th row

Problem/Recommendation: “Publication date.”

Rationale: Missing period.

US_E056:

Note 2 after Table 7.10

Problem: “In the case of Earth-fixed”

b. “The Earth-fixed approximations”

also sentence after Figure 7.12.

Recommendation: Add missing hyphen. See Binding in Table 7.10, where it is hyphenated.

US_E057:**7.5.1, 2nd paragraph after Note 1, 2nd sentence****Problem/Recommendation:** “of the first point of Aries”

also sentence before Note 2

“of the first of point of Aries”

Rationale: Typo. See 1st sentence after Note 1.**US_E058:****1st paragraph after Figure 7.7, last sentence****Problem/Recommendation:** “which varies as a function of time.”**Rationale:** Missing word.**US_E059:****7.5.5,****Problem/Recommendation:** The ~~heliocentred~~ *heliocentric planet ecliptic dynamic binding category*”**Rationale:** See “Table 0.1 — Heliocentric planet ecliptic dynamic binding category”.**Clause 8****US_E060:****Clause 8.1, first paragraph, third sentence:****Problem:** The second word “some” should be deleted.**Recommendation:** In some cases, ~~some~~ the CS parameters are bound by ORM parameters.**Rationale:** Clarity.**US_E061:****Table 8.30, Label - SRF_GEOCENTRIC_EARTH_1984, Region****And Label - SRF_GEODETTIC_EARTH_1984, Region****Problem:** “Global Earth.”**Recommendation:** Change to “Earth, Global” and all such occurrences.**Clause 9****US_E062:****Clause 9, Table 9.2:** Provide reference for VOS_NAVD_1988.**Change:** Add reference for NAVD. Report cover page is attached.

"NAVD Cover.pdf"

Rationale: Completeness**Clause 10****US_E063:****Clause 10.1, first paragraph, fourth sentence:****Problem:** Comma is missing.**Recommendation:** Next, the general case of changing the SRF representation of a spatial position is specified, followed by important special cases.**Rationale:** Clarity and consistency.**US_E064:****Sentence after equation 10.2****Problem/Recommendation:** “when neither the source nor the target ~~are~~ is necessarily”

Rationale: Subject-verb agreement.

US_E065:

Clause 10, Paragraph 10.3.1:

Problem: Incorrect subscript for third reference to an ORM in second sentence.

Recommendation: Change third ORM_S in the second sentence to ORM_R .

Rationale: Correctness

US_E066:

Clause 10, Paragraph 10.3.1:

Problem: Incorrect cross-reference in third sentence.

Recommendation: Change “Equation (7.3)” to “Equation (7.5)” in the third sentence.

Rationale: Correctness

US_E067:

Clause 10, Paragraph 10.3.1, Figure 10.1:

Problem: Mislabeled transformation.

Recommendation: Arrow from ORM_R to ORM_T should be labeled H^{-1}_{TR} .

Rationale: Correctness

US_E068:

Sentence after equation 10.8

Problem/Recommendation: “Note that rotation matrix T_{ST} may be determined directly”

Rationale: Missing word.

US_E069:

Sentence after equation 10.9

Problem/Recommendation: “As a consequence, if the difference (Equation (10.9)) between the specified rotation parameters ORM_S and ORM_T are sufficiently small, Equation (7.4) (or equivalently, the Bursa-Wolfe equation (see Annex B) applies.”

Rationale: Missing right paren. Suggest the sentence be rewritten for clarity. Suggestion is to remove the parenthesis before “or” and put a comma before “or” and remove the comma after “equivalently”. Also, check the directives and editorial rules for setting off the equations that have links.

US_E070:

10.3.3, last sentence

Problem/Recommendation: “when ~~a~~ an ORM”

Rationale: Agreement.

US_E071:

Note before equation 10.15

Problem/Recommendation: “In the literature, a transformation”

Rationale: Missing comma.

US_E072:

Sentence after equation 10.19

Problem/Recommendation: “In the case for which the SRFs are not both object fixed and/or are for different spatial objects”

Rationale: Remove extra space after “fixed”.

US_E073:

Paragraph before 10.4.2

Problem/Recommendation: “(or ~~a~~ an augmented map”

Rationale: Agreement.

US_E074:

Sentence after equation 10.31

Problem/Recommendation: “Furthermore, if”

Rationale: Missing comma.

US_E075:

Sentence after equation 10.32

Problem/Recommendation: “is equivalent ~~the~~ to the promotion operation”

Rationale: Typo.

US_E076:

Sentences before and after equation 10.33

Problem/Recommendation: “~~celestiodetic~~ celestiodetic”

Rationale: Misspelling.

US_E077:

Sentence after Figure 10.3

Problem/Recommendation: “Curvilinear CSs do not have a spatially linear vector space structure so ~~that~~ there ~~is~~ is no apparent way to specify a direction with curvilinear coordinates.”

Rationale: Typos.

US_E078:

Sentence after Figure 10.3

Problem : The use of the word “apparent” does not appear to be the proper word for an international standard.

Recommendation: Select another way to better capture the intended meaning of the sentence without using the word “apparent”.

US_E079:

10.5.1, 2nd last paragraph, 2nd sentence

Problem/Recommendation: “In the case of curvilinear SRFs that ~~is~~ are not based on oblate spheroid ORM realization, directions shall be specified as a normalized vector in the celestiocentric SRF instance based on the same ORM, with a nominal celestiocentric reference position of $\mathbf{c} = (0,0,0)$.”

Rationale: Missing word and number agreement is off.

US_E080:

10.5.1, last paragraph

Problem/Recommendation: “SRF based on ~~a~~ an oblate spheroid”

Rationale: Agreement.

US_E081:

10.5.2, 2nd paragraph, 1st sentence

Problem/Recommendation: “to an ~~LPT~~ LTP SRF” Remove extra space between acronyms.

Rationale: Extra space and misspelling.

US_E082:

10.5.2, 2nd paragraph, last sentence

Problem/Recommendation: “points in the directions of the ORM”

Rationale: Number agreement.

US_E083:

10.6.3, 2nd sentence after table

Problem/Recommendation: “short length case ease”

Rationale: Double word error.

US_E084:

10.6.3, 5th sentence

Problem/Recommendation: “OBS are exactly ~~antipodal~~ antipodal”

Rationale: Misspelling.

US_E085:

10.7.2.1, 4th sentence

Problem/Recommendation: “and may be specified as a function”

Rationale: Missing word.

US_E086:

10.8, Example 2nd sentence

Problem/Recommendation: “abstract models are rotated or otherwise transformed by a unitary operator”

Rationale: Incorrect verb tense.

US_E087:

10.8, 5th sentence

Problem/Recommendation: “to the position-space of ~~and~~ an ORM for another object”

Rationale: Typo.

Clause 11

US_E088:

11.1, 5th paragraph end

Problem/Recommendation: “~~implementatuion~~ implementation”

Rationale: Misspelling.

US_E089:

11.1, last sentence

Problem/Recommendation: “The API specifies a set of concrete classes”

Rationale: Missing word.

US_E090:

Clause 11.2.3, first paragraph, third and fourth sentences:

Problem: Commas are missing.

Recommendation: If two Object_References are equal, they refer to the same object instance. If an Object_Reference is equal to the special value NULObject, it does not reference any object instance.

Rationale: Consistency.

US_E091:

11.2.6.1, 3rd and 4th sentences

Problem/Recommendation: “The first set of non-object structured data types collects the parameters needed to specify an SRF. The second set ~~is~~ collects the coordinate components”

Rationale: Number agreement.

US_E092:

11.2.6.3.1 and 11.2.6.3.2

Problem: Cartesian spelled wrong 3 times.

Recommendation: Needs to be capitalized also (2 times).

Rationale: Misspelling. Capitalization per NSOED.

US_E093:**11.2.6.3.2, 11.2.6.3.3, 11.2.6.3.4**

Problem/Recommendation: “for a 3D eCartesian SRFs”
“for a 2D spherical based SRFs”
“for a spherical based SRFs”

Rationale: Number agreement error. See 11.2.6.3.1.

US_E094:

Problem/Recommendation: “The SRF objects ~~specifies~~ specify”

Rationale: Number agreement error.

US_E095:**11.3.1, 2nd paragraph, 1st sentence**

“of ~~sub-classes~~ subclasses”

Rationale: Since we use superclass as one word, subclass should be one word also. NSOED.

US_E096:**Table 11.7, SRF3Dbase, semantics**

Problem/Recommendation: “and for a specific SRF creates a direction ~~instances~~ instance initialised with the values passed in.”

Rationale: Number agreement error.

US_E097:**Table 11.7, GetCoordinate3DValues, error conditions 2**

Problem/Recommendation: Also GetCoordinate2DValues

“not initialised through the ~~AP~~ API, or is not in the source_srf.”

Rationale: Typos.

US_E098:**Table 11.7, ChangeDirectionSRF, semantics**

Problem/Recommendation: Missing period at end of sentence.

Rationale: Punctuation.

US_E099:**Table 11.7, last row, last item**

Problem/Recommendation: “EXTENDED_DESTINATION, if the coordinates was changed to the target_srf, but it is now in the extended region of target_srf.”

Rationale: Number agreement error. See error condition #4.

US_E100:**Table 11.8, Create2DCoordinate, semantics**

Problem/Recommendation: “creates a 2D coordinate ~~instances~~ instance initialised with the values passed in.”

Rationale: Number agreement error.

US_E101:**Table 11.8, Create2DCoordinate, semantics**

Problem/Recommendation: Add final period.

Rationale: Missing.

US_E102:

Table 11.8, last Abstract operation, Semantics

Problem/Recommendation: “associated with ~~an~~ a 2D coordinate.”

Rationale: Typo.

US_E103:

11.3.3, 3rd sentence

Problem/Recommendation: “It adds operations for the surface CS ~~induces~~ induced on the zero ellipsoidal height surface.”

Rationale: Typo.

US_E104:

11.3.3, 4th sentence

Problem/Recommendation: “It also adds ~~a~~ an operation to create”

Rationale: Typo.

US_E105:

Table 11.9, 1st input and 2nd output

Problem/Recommendation: “~~seond~~ second”

Rationale: Typo misspelling.

US_E106:

Table 11.10, 1st semantics

Problem/Recommendation: Add period to end of phrase.

Rationale: Missing final period.

US_E107:

Table 11.20, 1st semantics

Problem/Recommendation: “~~polar~~ polar”

Rationale: Typo misspelling.

US_E1108:

Table 11.22, semantics

Problem/Recommendation: “Creates ~~a~~ an equatorial inertial”

Rationale: Agreement.

US_E109:

Table 11.33, semantics

Problem/Recommendation: “Creates a ~~L~~Lambert”

Rationale: per NSOED.

US_E110:

Table 11.35, semantics

Problem/Recommendation: “Creates ~~a~~ an equidistant”

Rationale: Agreement.

US_E111:

Table 11.37, semantics

Problem/Recommendation: “Creates ~~a~~ an Alabama SPCS”

Rationale: Agreement.

Clause 12

US_E112:

12.1, 2nd sentence

Problem/Recommendation: “The membership of each set may be extended”

Rationale: Missing word.

US_E113:

12.2.2 2nd paragraph, item d.

Problem/Recommendation: Insert hyphen between “human” and “readable”

Rationale: These words are intended to be in tandem to describe a characteristic and as such the use is considered a compound adjective. It is noted that elsewhere in this standard, this use is hyphenated.

US_E114:

12.2.5.1 3rd sentence

Problem/Recommendation: Use the title of the subclause rather than just the number. Check the directive regarding this matter to see if beginning sentences with Arabic numbers is allowed.

Rationale: It is not considered good practice to begin sentences with Arabic numbers.

US_E115:

12.2.5.2 Paragraph 1, item d, 4th sentence

Problem/Recommendation: Break the sentence into two. If this action is not taken, then it is suggested that a semicolon be used in lieu of the comma after the word “reference”.

Rationale: Ease of understanding. The sentence is unnecessarily lengthy.

US_E116:

12.4 p

Problem/Recommendation: Add final sentence period.

Rationale: Missing.

US_E117:

12.7.5, 1, ii, 2nd sentence

Problem/Recommendation: “or registered it must ~~be~~ first be registered before”

Rationale: Word order error.

Clause 13

US_E118:

13.1, 1st paragraph, 2nd sentence

Problem/Recommendation: make p in “Profile” lower case

Rationale: SRM Profile is not a proper name.

US_E119:

13.3 2nd paragraph, 1st sentence

Problem/Recommendation: Delete “prepend or”. Or if it is felt it is needed, an alternative is to delete “prepend or” and insert “the beginning or end of” between “to” and “labels”.

Rationale: This usage is not supported by the dictionary.

Annex A

US_E120:

A.2, 2nd sentence

Problem/Recommendation: “The set of all n -tuples of real numbers, is denoted by \mathbf{R}^n .”

Remove comma.

Rationale: Comma usage is incorrect here.

US_E121:

A.4 Smooth functions on \mathbf{R}^n , item c

Problem: Missing article.

Recommendation: Modify as indicated.

“...evaluated at a point in...”

US_E122:

A.4 c

Problem/Recommendation: “evaluated at a point in the domain”

Rationale: Missing word.

US_E123:

A.6.2, paragraph before note

Problem/Recommendation: “When $a < b$, the surface is called ~~an~~ a *prolate spheroid*.”

Rationale: Agreement.

Annex C

US_E124:

C.1, 5th sentence

Problem/Recommendation: “~~eoponents~~ components”

Rationale: Misspelling.

US_E125:**C.1, 6th sentence**

Problem/Recommendation: “Three examples of ORMs based on an ORM template ~~is~~ are shown in **Error!**
Reference source not found..”

Rationale: Subject-verb agreement. Examples is the subject.

Annex E**US_E126:****Table E.4, Label ORM_ADINDAN_ETHIOPIA, last column**

Problem/Recommendation: ~~Ethiopa~~ Ethiopia

Rationale: Misspelling.

US_E127:**Table E.4, Label ORM_EUR_1950_CHANNEL_ISLANDS, references**

Problem/Recommendation: [HELM, "EUR", "~~Chanel~~ Channel Islands"]

Rationale: Misspelling.

US_E128:**Table E.4, Label ORM_MERCHICH, region**

Problem/Recommendation: ~~Morreee~~ Morocco

Rationale: Misspelling.

US_E129:**Table E.4**

Problem/Recommendation: Reformat table to have one or two lines/ORM with repeating table headers from one page to the next.

Rationale: Brevity

US_E130:**Table E.4**

Problem/Recommendation: Reformat table to have one or two lines/ORM with repeating table headers from one page to the next.

Rationale: Brevity

US_E131:**Table E.4**

Problem/Recommendation: Do not allow information associated with a specific ORM to be split across pages.

Rationale: Improved readability when using hardcopy of document.

US_E132:

Table E.4: References for NAD 1983 are incomplete.

Problem/Recommendation: Add reference to NAD 1983 report. See related comment for Bibliography. Report cover page is attached.



NADof1983
Cover.pdf

Rationale: Completeness

US_E133:**Annex E: ORMS**

Problem/Recommendation: Reference for ORM_WGS_1972 is Seppelin, T. O.; The Department of Defense World Geodetic System 1972; Technical Paper; Defense Mapping Agency; Washington, DC; May 1974 and 83502T (for binding to WGS 84). Change reference for ORM_WGS_1984 to 83502T.

Rationale: Reference listed does not provide information about ORM_WGS_1972.

Annex F**US_E134:****Table F.4**

Problem/Recommendation: “~~provisional~~ provisional”

Rationale: Misspelling.

US_E135:**Table F.4**

Problem/Recommendation: “~~istema~~ sistema de referencia geocentrico para america del sur

Rationale: Capitalization error.

Annex G**US_E136:****G.1**

Problem/Recommendation: Add space between G.1 and Introduction.

Rationale: Missing.

US_E137:**G.2, 1st sentence**

Problem/Recommendation: Add space between “Redefinition” and “of”.

Rationale: Space missing.

US_E138:

G.2, 3rd paragraph

Problem/Recommendation: “To ~~insure~~ ensure stable evolution”

Rationale: Incorrect word choice.

US_E139:

G.3.1 b

Problem/Recommendation: “~~re-used~~ reused”

Rationale: NSOED.

Annex H

US_E140:

H.4

Problem/Recommendation: “Proposal for the registration of ~~a~~-an RD”

Rationale: Consistency with H.7 – “Proposal for the registration of an SRFT”.

US_E141:

H.9, last sentence

Problem/Recommendation: “In the case that the SRF set membership is specified through an explicit parameterized list of members, the following form”

Rationale: Comma rules.

Annex J

US_E142:

Table J.7, ORM_N_AM_1927_EASTERN_US, Description

Problem/Recommendation: “Eastern United States (Alabama, Connecticut, Delaware, District of Columbia, Florida, Georgia, Illinois, Indiana, Kentucky, Louisiana, Maine, Maryland, Massachusetts, Michigan, Minnesota, Mississippi, Missouri, New Hampshire, New Jersey, New York, North Carolina)”
Was more of this list left off?

Rationale: Spell out N. Carolina in full since the other states are. Add final parenthesis.

US_E143:

Alphabetical Index

Lambert, 2 entries

Problem/Recommendation: Both should be capitalized since Lambert is a proper noun.

Rationale: Consistency (one is capitalized and the other is not).

Bibliography

US_E144:

83582

Problem/Recommendation: “[online]. 1st ed.. Washington:” Remove 2nd period after “ed”.

Rationale: Typo.

US_E145:

Bibliography: Add entry for WGS 72 report.

Problem/Recommendation: Add WGS72: Seppelin, T. O.; The Department of Defense World Geodetic System 1972; Technical Paper; Defense Mapping Agency; Washington, DC; May 1974 to the table.

Rationale: Completeness.

US_E146:



Bibliography: Add entry for North American Datum 1983 Report

Problem/Recommendation: Add NAD83: Schwarz, Charles R. (ed), NOAA Professional Paper NOS 2, *North American Datum of 1983*, National Geodetic Survey, Rockville, MD; December 1989, U.S. Department of Commerce, National Oceanic and Atmospheric Administration to the table.

UK National Body Comments on ISO/IEC 18026 WD7

General

UK_G001:

Entire document.

All concepts should be defined before they are used. In particular, the abstract classes listed in 11.1 should be defined before they are used.

UK_G002:

Entire document.

The word “initialized” is misspelled as “initialised”. Do not use the Microsoft British dictionary to verify this. It is wrong. The SOED specifies the correct spelling of this and other “ize” suffix words as using a zed. All occurrences of such words in the text should be identified and changed where necessary to use the “ize” suffix.

UK_G003:

Entire document

When tables are divided into major and minor rows, page breaks should only occur on major row boundaries. See Table E.4 for an example.

UK_G004:

Entire document

Now that ISO 19111:2003 Geographic information - Spatial referencing by coordinates is published, concepts in the SRM need to be consistent with and derived from those in ISO 19111. Note that the scope statement of ISO 19111 defines an area of applicability that includes the SRM:

“This International Standard defines the conceptual schema for the description of spatial referencing by coordinates. It describes the minimum data required to define one-, two- and three-dimensional coordinate reference systems. It allows additional descriptive information to be provided. It also describes the information required to change coordinate values from one coordinate reference system to another.

This International Standard is applicable to producers and users of geographic information. Although it is applicable to digital geographic data, its principles can be extended to many other forms of geographic data such as maps, charts, and text documents.”

ISO standards may not conflict unless they are intended for totally different applications. We do not believe that this is the case with SRM and ISO 19111.

Definitions that are different in SRM from ISO 19111 can be moved to footnotes that say that some communities use the following alternative definition and that explain how the ISO 19111 definition used in the SRM is related to the non-standard definition in the footnote.

At a minimum, the following concepts from ISO 19111 now need to be used in SRM instead of the present SRM concepts and ISO 19111 needs to be cited as the source of the definition:

coordinate

one of a sequence of n numbers designating the position of a point in n -dimensional space

compound coordinate reference system

coordinate reference system using two other independent coordinate reference systems to describe a position

coordinate conversion

change of **coordinates**, based on a one-to-one relationship, from one **coordinate system** to another based on the same **datum**

coordinate operation

change of **coordinates**, based on a one-to-one relationship, from one **coordinate reference system** to another

coordinate reference system

coordinate system that is related to the real world by a **datum**

coordinate system

set of mathematical rules for specifying how **coordinates** are to be assigned to points

coordinate transformation

change of **coordinates** from one **coordinate reference system** to another **coordinate reference system** based on a different **datum** through a one-to-one relationship

datum

parameter or set of parameters that serve as a reference or basis for the calculation of other parameters

easting

E

distance in a **coordinate system**, eastwards (positive) or westwards (negative) from a north-south reference line

ellipsoid

surface formed by the rotation of an ellipse about a main axis

ellipsoidal height**geodetic height**

h

distance of a point from the **ellipsoid** measured along the perpendicular from the **ellipsoid** to this point positive if upwards or outside of the **ellipsoid**

geodetic coordinate system**ellipsoidal coordinate system**

coordinate system in which position is specified by **geodetic latitude**, **geodetic longitude** and (in the three dimensional case) **ellipsoidal height** as positive

geodetic datum

datum describing the relationship of a **coordinate system** to the Earth

NOTE: In most cases, the geodetic datum includes an ellipsoid definition.

height

h, H

distance of a point from a chosen reference surface along a line perpendicular to that surface

map projection

coordinate conversion from a **geodetic coordinate system** to a plane

polar coordinate system

coordinate system in which position is specified by distance and direction from the origin

projected coordinate system

two-dimensional **coordinate system** resulting from a **map projection**

spatial reference

description of position in the real world

vertical datum

datum describing the relation of gravity-related **heights** to the Earth

As an alternative to implementing this comment, if agreement can be reached with ISO TC 211 to amend ISO 19111 to change any of its present concepts to use the ones in SRM instead, then the present SRM concepts may be retained.

UK_G005:**Entire document**

The set of concepts in the SRM should be reduced to only what has been implemented and proven in practice. In particular, no concept - and in particular, no mathematical operations -- not implemented in API developed by the SEDRIS Organization should be included in the standard. The SEDRIS Organization should be asked to present the current state of its API implementation at the meeting where these comments are discussed and the full source should be made available for try-out and inspection by others.

This comment was previously part of UK_G003 on SRM WD 7 that was discussed and accepted in Stuttgart. This comment said in part:

“UK G003

The coordinate systems, reference datums, and SRFs should be reduced to include only those in which the active participants in the development of this IS have direct and applicable implementation experience. The scope should be restricted to what has already been implemented and proven in practice by the SEDRIS Organization in its SRM API.”

It is accepted practice within SC 24 that all standards must be proven in practice before they are allowed to advance to FCD. It would be acceptable to issue a next version of SRM at CD to allow additional time to demonstrate the correctness of the algorithms in the SRM through their implementation. Then concepts not demonstrated and proven can be removed.

UK_G006:**Entire document**

Most RDs and other concepts related to non-Earth objects should be removed from the SRM. The exception is those concepts needed to support important solar and earth coordinate systems. This comment relates specifically to Annex D, hence is given in detail by UK T093.

UK_G007:**Entire document**

The agreed response to comment UK G003 on the SRM WD 7 (last version before CD) has not yet been implemented. This comment said in part:

“UK G003

The coordinate systems, reference datums, and SRFs should be reduced to include only those in which the active participants in the development of this IS have direct and applicable implementation experience. The scope should be restricted to what has already been implemented and proven in practice by the SEDRIS Organization in its SRM API.”

The response stated:

“Response: There are two sets of concepts that are of concern. One is RDs for non-earth objects and the other is SRFs in 8.4.12 through 8.4.20. Establishing liaison with the appropriate international organizations and asking them to review the next version of this document solves the problems that this comment addresses. The next version should add planetodetic CS and planetodetic SRF.

There is a problem in dealing with parameters that are published by others and that may change over time. We need a unique way of designating a set of parameters (by code and label). We also need to be able to specify the epoch of validity of a parameter and to allow later parameters to supercede earlier ones.”

The appropriate international organizations have not been contacted and they did not review the SRM CD. Therefore, the offending material should be removed.

We know of no way that the issue in the second part of this response can be handled within an International Standard due to the long time frames that it takes to amend such a standard. Instead, the proper place for such parameters is in a document that can change more frequently, such as a register or a document controlled and revised by an organization (such as IAU/IAG Working Group on Cartographic Coordinates and Rotational Elements of the Planets and Satellites) with the technical expertise to update the parameter values.

UK_G008:

Entire document

Careful consideration should be given to ceasing development of this material as an International Standard and instead publishing it as a Technical Report or a Technical Specification. The main reasons that such a change of document form should be considered are:

1. the lack of maturity of the material as demonstrated by the evolution of material during the working draft process and the amount of new material seen in this CD for the first time;
2. the lack of implementation and use of the concepts in the SRM in practice in the general community outside of military modeling and simulation, and in particular by the OGC and by TC 211 standards; and
3. the lack of international consensus as demonstrated by how the SRM is not harmonized with International Standards developed by TC 211 or with the commercially-important work of the OGC, and particularly with GML.

UK_G009:

Entire document

All figures should be re-drawn in a format that may be edited in MS Word. This should be done before the next version is published. This was agreed to in Stuttgart in response to UK G005 on WD 7, but was not implemented as agreed. That comment and response are reproduced below.

“All figures must be re-drawn so they can be edited. Most are in a special Word 98 editor that the integrating editor cannot open and modify. This made it impossible to make many needed changes to SRM WD 7. Further, WG 8 needs to establish a policy that only Microsoft Word Pictures (Insert / Object / Microsoft Word Picture) can be used for all line art drawings in all its standards. If there is an image that is needed, JPEG or GIF Formats should be used (Insert / Picture / From file...). Other formats should not be used.

Response: Accepted in principle; specification of mechanisms for achieving consistency in the creation and management of diagrams are left to the discretion of the Editor.”

Without this change not only is it hard to comment on changes to figures but the ITTF editors will have great trouble modifying figures for consistency with ISO style and for proper reproduction in PDF format.

UK_G010:**Entire document**

The font and point size in text created in MathType and included in figures does not match that of text created in MS Word. For examples, look at Table 5.13 and the figure in Table 5.25. These need to be made consistent throughout by changing the set-up of MathType and the text-in and method of generation of figures as required.

Technical**Table of Contents****UK_T001:****Entry for Clause 3.**

There is no hyperlink to Clause 3.

UK_T002:**Table of tables.**

This table of tables is incomplete. There are many missing tables including all in Annex E. If there is to be a table of tables, it should be complete.

Foreword**UK_T003:****Throughout**

Changes corresponding to those made for the EDCS FDIS should be made in this material also.

Introduction**Clause 1—Scope****Clause 2—Normative references****UK_T004:****Throughout**

Changes corresponding to those made for the EDCS FDIS should be made in this clause also.

Clause 3—Terms, definitions, symbols and abbreviated terms**Clause 4—Concepts****Clause 5—Coordinate systems****UK_T005:****5.2.5.1**

The document referenced in the footnote is not listed in the bibliography or normative references and the citation is not in the proper style:

“Geographic information -Spatial referencing by coordinates (DIS 19111) defines meridian as intersection between an ellipsoid and a plane containing the semi-minor axis of the ellipsoid. That definition includes the antipodal meridian and the pole points.”

UK_T006:

5.3.3.1, last sentence and footnote.

The sentence “No map projection CS can eliminate all distortion⁸.” adds no normative value and should itself either be the subject of a note or should be removed. The footnote 8 “The proof of this assertion is beyond the scope of this document.” should be backed up by a reference to a text where this is proved.

UK_T007:

5.3.4, last paragraph.

There is a missing figure that should be supplied.

UK_T008:

5.4. throughout.

Nowhere is it explained how the labels for the various CS specifications are formed. For example, what is the meaning of “S3” in “CS_S3_SURFACE_GEODETTIC”? An explanation of how labels are created should be provided to allow for consistent choice of labels during registration. As it is, the use of abbreviating terms in labels seems hopelessly inconsistent. As another example, some labels start with “CS_3D” while others start with “CS_MP”. It would seem more consistent to use something of the form:

CD_fntype_cstype_description.

This would at least make everything consistent.

UK_T009:

5.4, Table 5.7

This table should include the official labels for the coordinate systems. As it is, there is no place where such labels are gathered together. If it is decided that this table would be too busy with such labels, it is suggested that there be annexes containing indexes for the various categories of items listed alphabetically by label.

UK_T010:

5.4, Table 5.14, Field CS parameters and constraints

There is a disagreement in form and parallel structure in:

“ $a > b$: (oblate spheroid)

$a = b$: (spherical)”

between “oblate spheroid” and “spherical”. Using “spheroid” instead of “spherical” would be one solution.

UK_T011:

5.4, Table 5.19 and Table 5.23 Field Coordinates

Remove “or range” as it is only confusing in this context because this is an abstract coordinate system.

UK_T012:

5.4, Table 5.28 and Table 5.29, Field Figures

The figure does not stand alone with no explanatory text.

Clause 6— Temporal coordinate systems

Clause 7—Reference datums, embeddings and object reference models

UK_T013:

7.4.4, Table 7.9, ORMT label column

Some of the labels have inappropriately placed hyphens.

UK_T014:**7.4.4, Table 7.9, ORMT_3D_TRI_AXIAL_SPHEROID**

The content of the ORMT specification cell is missing.

UK_T015:**7.4.5, 7th paragraph, first sentence**

The language used here and throughout to describe specifications needs to be consistent. Rather than “The elements of an ORM specification are defined in Table 7.10” say “The fields of an ORM specification are defined in Table 7.10”. That is, consistently say that a specification consists of set of fields.

UK_T016:**7.4.5, Table 7.10 as well as most other tables in Clause 7 and Annex E**

Having a field “Date published” conflicts with giving a reference in the “References” filed. Other concerns are:

1. When is the date published not the date of publication of the reference?
2. What aspects of the specification were “published” on the “Date published”? Surely not the label and code?
3. If there was indeed a publication of the given date, should a reference to that publication not be given?

We believe that this field is best eliminated unless its definition (and that of other fields, if required) are modified to answer the above questions.

UK_T017:**7.4.5, Table 7.10 as well as most other tables in Clause 7 and Annex E**

The present region field values are not always meaningful. For example, in Table E.3 what does the region “Guinea-Bissau” mean? Is Bissau a part of Guinea? Or another administrative division that is added to Guinea to make up the region? The meaning *****

UK_T018:**7.4.5, Table 7.10 as well as most other tables in Clause 7 and Annex E**

We recognize that the “region” specifications may be to historical entities as they existed at the time of the publication of a specification. However, some of these specifications have various problems that we illustrate by example:

1. “FRG” in ORM_POTSDAM is jargon and not a country name or a country code.
2. Some names seem to refer to modern countries, yet they are not the correct UN and ISO names for those countries. Example: “Russia” instead of “Russian Federation” in ORM_SOVIET_GEODETIC_1985
3. Some region specifications are ambiguous. For example, what does “Spain (except Northwest)” mean in ORM_EUR_1950_SPAIN_EXCEPT_NW?

The way that Table 7.10 defines the Region field is: “Description of the subset of object-space to which the model applies.” However the contents of this field seems to be a much less precise thing, that is, only a general indication of an area where the model might apply. And this “general indication” seems to also sometimes be in terms of countries and their boundaries as they existed at some time in the past. All this needs to be clarified.

UK_T019:**7.4.5, Table 7.10 as well as most other tables in Clause 7, Annex E, Annex D and other locations where there is a “Description” field in specifications**

The Description field is defined as “Description of the ORM including name as published or as commonly known.” Based upon inspecting the contents of this filed in many tables, this field seems to usually contain just a published or common name. Further, the reader is left to guess whether the “name as published”

matches any names in the cited references. We believe that the name and definition of this field would be best changed to better reflect its actual usage. Our suggestion:

Published name Name(s) given to the concept embodied in this ORM in the reference(s).

In a few cases, extra data can be provided as a NOTE in this field. Example; the description “NTF, with the zero meridian set at Paris” might be handled partly as a note depending on the actual published name.

Clause 8—Spatial reference frames

UK_T020:

8.5.1, Table 8.3

The SRF name “Local space rectangular” is jargon and makes a distinction not made in similar SRF names. For example, in 2D SRFs, Azimuthal and Polar are both just as “local” and as “space”-related as “Local space rectangular” yet the words “local space” are not prepended to their names. For consistency the names should be either:

Rectangular, Azimuthal, and Polar

or

Local space rectangular, Local space azimuthal, and Local space polar.

We favor the former for simplicity.

Similarly the names of 3D SRFs “Local space rectangular” and “Local tangent plane: fail to capture the essence of the concepts and may even mislead because “Local tangent plane” sounds 2D not 3D. Better names should be found.

Clause 9—Vertical offset surfaces

UK_T021:

9.3, Table 9.2

The vertical offset surface identified by code 5 is missing.

UK_T022:

9.4

This non-normative material should be in a NOTE.

Clause 10—Spatial operations

UK_T023:

Throughout

The wording in this clause is considerably less polished than that of earlier clauses. There are many grammar and style mistakes and examples of informal and imprecise wording. Here is one example from 10.7.1:

“Understanding the definition of azimuth is important for coordinate conversion and other spatial operations.

On a sphere, a geodesic between points p1 and p2 is an arc of a great circle connecting p1 and p2. The problem of computing the angles of a spherical triangle given its vertices can be solved in closed form but approximations to the exact geodesic azimuth may be used to reduce processing time or to otherwise simplify the computation. In the general case of an oblate spheroid, the problem of computing the angles of an elliptical triangle usually does not have a closed solution. Several different approximations are commonly used. Note that in the definition of geodesic azimuth both points must be on the surface of the oblate spheroid (or sphere). If one or more of the two points are not on the surface, the effect of a non-zero ellipsoidal height must be accounted for [RAPP1] [RAPP2]. The geodesic azimuth geometry is depicted for point pairs in both hemispheres in Figure 10.4.”

This paragraph contains a sentence that is improperly in bold, notes that are not in proper style, a long digression into implementation considerations that is totally inappropriate, and many other transgressions.

Another example, from 10.7.5.2:

“The oblique Mercator map projection is conformal so that the scale factor and point scale are the same and are given by...”

The “so that” is not good style. A better way to say it is: “Because the oblique Mercator map projection is conformal, the scale factor and point scale are the same and are specified by...”

Clause 10 should be thoroughly reviewed and revised by an ISO-experience editor.

Clause 11— Application program interface

UK_T024:

11, Throughout

The parameter names of the object Create methods should agree with the field names of the matching SRF_Parameter data types. Note the difference between these for 3DLocalTangentPlane.

UK_T025:

11, Throughout

The API does not give the access desired to spatial operations at the level they are described in Clause 10. Clause 10 describes how operations are defined and composed to do specific things (like change SRF representation). The SRM API should have a "low-level" API piece that allows both access and specification at this level of detail. This is the only way, for example, that the API can be "extended" by a user to meet his or her needs without going through a long and complex process of registering new SRFs, defining new classes (by registration again), and finally waiting for some implementer of the SEDRIS API to support the newly registered items.

To be "complete", we expect the API to implement all of the operations defined in Clause 10. Only coordinate conversion is done now.

UK_T026:

11, Throughout

The API fails at being easy to use. If a user just wants to "convert some coordinates", they should be able to do this in a single function call that handles all the details and figures out all the un-supplied parameters from context, as much as possible. Such an "easy to use" API was a major design goal articulated in the past, but it seems to have been lost. The API needs to be re-designed to include an aspect that addresses high-level ease of use, as well as an aspect that addresses “low-level”, more complete and detailed control of the chain of operations to be applied.

UK_T027:

11, Throughout

It is unclear whether the "one class per SRF" (or is it SRFT -- the API actually has examples of both!) approach is a good one. First, SRFTs are just specification conveniences that you would not expect to see expressed in the API. It is the SRFs that should be of interest.

Given this observation, note that some SRF sets have many members.

SRFS_GTRS_GLOBAL_COORDINATE_SYSTEM has 49,896 members.

SRFS_UNIVERSAL_TRANSVERSE_MERCATOR has 120 members. Clearly the API can't have one class per SRF, so (if the API continues with this approach) there must be a conceptually clean way to distinguish SRFs that have their own classes from those classes whose instances implement one of a set of SRFs (with the set member to be implemented defined as an attribute of, or input parameter to, the creation operation that makes the class instance).

In the redesign of the API, serious consideration should be given to collapsing back down to a single class for each major group of SRFs (such as 3D ones), with the exact SRF being an attribute of the class. This would give what some commentators in the past have often ask for - **an operation that can create any SRF and that takes as an argument a variant record**. This design approach would also avoid the very messy confusion caused by making class names out of parts of labels -- the labels would be attributes and parameters, as they should be, and not part of the function names!

UK_T028:

11, Throughout

The API should be reduced to only what has been implemented and proven in practice. The SEDRIS Organization should be asked to present the current state of its API implementation at the meeting where

these comments are discussed and the full source should be made available for try-out and inspection by others.

This comment was part of UK G003 on SRM WD 7 that was discussed and accepted in Stuttgart. This comment said in part:

“UK G003

The coordinate systems, reference datums, and SRFs should be reduced to include only those in which the active participants in the development of this IS have direct and applicable implementation experience. The scope should be restricted to what has already been implemented and proven in practice by the SEDRIS Organization in its SRM API.”

It is accepted practice within SC 24 that all standards must be proven in practice before they are allowed to advance to FCD. It would be acceptable to issue a next version at CD to allow additional time to demonstrate the correctness of the algorithms in the SRM through their implementation.

UK_T029:

11.1

How the API works is rather hard to comprehend, based on the current introductory text. This may be easily corrected by adding some introductory text and an example or two. It seems that the designers intend that the way that SRF conversion happens is that a programmer creates an instance of the target SRF and an instance of the source SRF and then asks the target SRF to do the conversion by invoking its

ChangeCoordinateNDSRF operation (N= 2 or 3).

UK_T030:

11.2.2, 2nd para, last sentence

The non-negative short integer data type should have names consistent with the usage in other related standards. The style that should be used has the center word specifying the fundamental type of data, the 1st word specifying the abstract length, and the 3rd word specifying the value modifier. Hence, “Short_Unsigned_Integer” should be “Short_Integer_Unsigned”.

UK_T031:

11.2.2, last para, 2nd sentence

The correct standard for specifying the format of floating point numbers is “IEC 60559:1989, *Binary floating-point arithmetic for microprocessor systems* (previously designated IEC 559:1989).”

UK_To32:

11.2.4.2 and 11.2.4.3

These two data types be combined into a single data type “Indication_Of_Direction” with the same enumerants as specified for each. The usage of this data type will indicate whether it is “up” or “forward” that is being specified.

UK_To33:

11.2.5.1

Selection data types in other standards are all of abstract type Integer. This should also be the case for SRM. Note that all programming languages of which I am aware, represent enumerated data types as Integer data types as well.

UK_T034:

11.2.5.2

All 490 entries should be specified here.

UK_T035:

11.2.5.2

Since no specific ordering is mentioned in the text, it is assumed that the ordering is by ordering


```
false_easting    Long_Float;  
false_northing   Long_Float; }
```

The amount of programming needed to assign values to structure members, and then pass the structure, is not much different from assigning a value to the separate arguments as they are passed. In fact, it is likely that an application programmer will recognize that these data items are related and will follow good programming practice and store them in a structure, whether or not we predefine one for him or her. Recognizing this, the API just makes it harder on the programmer, as he or she must "unwrap" what has carefully been "structured" just to pass it to the API.

Note that passing a structure to an object instance says nothing about how the object instance chooses to store its data internally. It seems as if the API designers still mistakenly feel that how arguments are passed affects how you store and use them internally.

UK_To42:

11.2.6.1, 1st sentence

This sentence implies that some of the specified structure non-object data types are arrays. However, there are no arrays specified. The text "arrays or" should be removed unless at least one array data type is specified.

UK_To43:

11.2.6.2.1

This subclause is nonsense. There is no need for a structure data type containing only a single field. That field can be used by itself.

UK_To44:

11.2.6.2.6

One data type is defined in 11.2.6.2.6. "Mercator_Parameters" for the parameters that correspond to SRFT_MERCATOR and SRFT_TRANSVERSE_MERCATOR, yet these take different parameters.

UK_To45:

11.2.6.2.6 and 11.2.6.2.7

The same term should be used for the central scale factor. It should that specified for OM_Parameters.

UK_To46:

11.2.6.2.6 and 11.2.6.2.7

The same parameter ordering should be used for both of these. Thus, central_scale_factor should either precede the false_xxx definitions or follow them in both specifications.

UK_To47:

11. Missing specifications

Where are the definitions of variant record types that allow for the specification of arbitrary SRF parameters and arbitrary SRF coordinates? These are essential for specifying a data type that can represent the values of SRF parameters and SRF coordinates when the SRF is not known ahead of time.

A suitable definition would be variant record types that use the SRF data type as a discriminant and then provides a definition for each possible variant. A value of a discriminant that does not

need further information would specify an “unused” variant of type Short_Integer. For SRF_Parameter, the variants would use the various “XXX_Parameters data types already specified. For SRF_Coordinate, the variants would use the various XXX_Coordinate data types already specified.

THIS IS A CRITICAL ISSUE.

UK_To48:

11.2.6.3, throughout

Data type names should not use a numeral as the initial character of the name since many programming languages do not allow this. It is suggested that the “2D”s and “3D”s be placed after the type of coordinate to be consistent with the placement of similar dimensionality information specified for various SRF parameter data types.

UK_To49:

11.2.6.3.3 and 11.2.6.3.4

These coordinates are not what are commonly referred to as “spherical coordinates”. They should be renamed “geodetic coordinates” instead.

UK_To50:

11.2.7.1, 1st sentence

This sentence makes no sense. Either something is a data type or an object type. It cannot be both. Since these are clearly object types, they should be called that and the entirety of 11.2.7 be moved to follow 11.3.2. In addition, each of the abstract object types should be accompanied by a set of concrete object types that define the representations for each version that can actually be instantiated. This was done for the abstract SRF class but not for any of the others.

UK_To51:

11.2.7.2

There should not be two different things with the same name. There is already an “SRF” non-object data type. Therefore, this object data type should use something else such as “SRF_Object”. It is suggested that all object data types should have the word “Object” as a name suffix to distinguish objects from non-objects.

UK_To52:

11.2.7.2

This object data type uses the class LifeCycleObject before it is defined. It is suggested that the detailed characteristics of LifeCycleObject be specified before this class is used elsewhere. The general description in the penultimate paragraph of 11.1 is inadequate. This applies in general. An object hierarchy diagram should be provided either in 4.10 or in 11.1.

UK_To53:

11.3, throughout

Each class be placed in its own subheading so that the TOC will contain, and therefore hyperlink to, these definitions. This will correspond to each data type being in a separate subclause.

UK_To54:

11.3, throughout

The parameter names are inconsistently formed. For example, Create3DCoordinate uses

“first/second/third_coordinate_component” while Create2DCoordinate uses “first/second/third_value”. All should be in the form used by Create3DCoordinate. This also applies to “direction”.

UK_To55:

11.3, throughout

Some parameter names have the access type appended. This information is redundant and should be removed. The row header for the parameter clearly specifies what the access type is.

UK_T056:

11.3 throughout

It is confusing that all possible returns are not listed in "Error conditions" in each class specification and that Status_Code is not an explicit output (or explicitly tied to "Error conditions"). A programmer that gets a status code and looks in the API spec will not see the code he or she has received necessarily listed there. We understand the desire to keep error condition return conceptually separate from output parameters, but the fact that these "errors" show up both in Status_Code and in "Error conditions with different "numbers" beside them will be confusing to a programmer.

There should be a "Status" for each operation that lists all possible returns for the operation as values of Status_Code, rather than separate error conditions and Status_Code.

UK_T057:

11.3 throughout

Attributes should be included as part of the object model, as the use of attributes in specifications avoids repeated operations like GetSRFParameters, GetDirectionValues, GetCoordinate3DValues and others.

Attributes of all objects can be set and inquired by using a common technique defined with the base object class. Data need only be declared in the specification in the list of attributes to automatically have the get and set operations defined.

Values of attributes can be set and inquired in different ways (including procedurally), as the API is bound to different object systems and programming languages.

UK_To58:

11.3.1, 1st para sentence

This paragraph is introductory to the entire clause and hence should be in 11.1.

UK_To59:

11.3.1

The major subclause is entitled “Object types” but everywhere the term “classes” is used. One of these terms should be used consistently and the other discarded. As it is, the entirety of 11.3.1 seems to not have anything to do with “Object types”. It is suggested that the term “Object classes” be used since this is common within the industry. Also, nowhere is there any real discussion of how these classes are used. There should be at least a description of the concept of an “Object instance”.

UK_To60:

11.3.1, 2nd para

There are many mentions of “functions” herein. However, the functionality made visible by interfaces to objects are typically called “methods” in the industry. The term “functions” should be used access to non-object functionality.

UK_T061:

11.3.1, second paragraph

It is unwise to require that errors be tested for and returned in a particular order. This forces implementers to follow certain internal design strategies and prevent the exploitation of parallelism. Specifically, this text in 11.3.1, second paragraph. should be removed: “The error conditions are listed in the descending order of

priority. The second condition can only be true if the first condition is not present, and so forth for the remaining error conditions.”

UK_To62:

11.3.1, last para, last sentence

This last clause of this sentence should be omitted. Language bindings should not be allowed to specify anything that would introduce functionality that would create divergent behaviour between implementations using different languages. Either the output values should be mandated by the abstract specification to be “undefined” or the exact values allowed should be specified.

UK_To63:

11.3.2, throughout

The headings for all abstract classes should be consistent. In some cases, “abstract class” is used to identify object classes that are abstract and in other cases, not (see `LifeCycleObject`). Since these are all in *11.3.2 Abstract classes* there is no need to label any of them as “abstract class”.

UK_T064:

11.3.2, throughout

The rows entitled “Abstract operation” should instead be “Method”.

UK_T065:

11.3.2.1, Create3DCoordinate

There should be an input parameter specifying the “specific SRF” mentioned in the semantics.

UK_T066:

11.3.2.1, Create2DCoordinate

There should be an input parameter specifying the “specific SRF” mentioned in the semantics.

UK_T067:

11.3.2.2, Free2DCoordinate

Why is there a method `Free2DCoordinate`? This is not needed since the same functionality is inherited from the `LifeCycleObject` interface.

The method should be removed.

UK_T068:

11.3.3

This specification of an abstract class is at the wrong heading level.

It should be demoted to be 11.3.2.3.

UK_T069:

11.3.3, CreateSurfaceCoordinate

The output parameter is inconsistently named when compared to the other `CreateXXXXCoordinate` methods. It should be named simply “coordinate”.

UK_T070:

11.3.3.1

The heading for `LifeCycleObject` is at the correct heading level but should be 11.3.2.4. This is result of the need to demote 11.3.3 to 11.3.2.3.

UK_T071:

11.3.3.1, Abstract operations

The parameter names used are not of the agreed form. They should be “object_reference”.

UK_T072:

11.3.3.2

The heading for SRF concrete classes should be under the heading for abstract classes. Instead this heading should be promoted to be 11.3.3. Moreover, this entire subclause should be reorganized as follows:

11.3.3 Concrete classes**11.3.3.1 Classes with no operations****11.3.3.2 SRF classes****UK_T073:****11.4, Table 11.40 and Table 11.41**

The RadianToDegree and DegreeToRadian functions are not needed. Not only are these defined in EDCS already, but these are built-in library functions on all systems. There must be a compelling reason to define special ones here and we see none.

UK_T074:**11.6, throughout**

These data types are actually selection data types and should be specified there.

UK_T075:**11.6, throughout**

All selections within the selection data type should be explicitly specified. It is not clear how to specify the items between 1 and last since there is only an implicit ordering in the earlier clauses and the means for specifying the labels is not provided anywhere.

UK_T076:**11.6, throughout**

Since the comments have been allowed to wrap to the left margin, it is nearly impossible to make sense of these data types. If comments are to be provided, they should have a hanging indent at the beginning of the comment.

UK_T077:**11.6.4, ORMs**

The data types in 11.6.4 and 11.2.5.2 are redundant. One should be removed.

UK_T078:**11.6.8, throughout**

Each of the set member data types should have its own subclause heading.

UK_T079:**11.6.8, throughout**

The appendage “SRFSM_” is redundant and should be removed. Instead, “SRFS_” should be replaced by “SRFSM_”.

UK_T080:**11.6.8, SRFS_UNIVERSAL_TRANSVERSE_MERCATOR_SRFSM_Code**

Only one label is provided for the Southern Hemisphere. Since it is labeled as “zone 60”, zones 1 – 59 and 61 – 129 seem to be missing for the southern hemisphere.

Clause 12— Registration**UK_T081:****Throughout**

This clause should be re-written, patterned on the corresponding clause in the EDCS. For example, the wording in 12.2.1 is awkward: “The specification of each SRM concept that is either a standardized concept or a registered concept shall include the following fields:” Better wording patterned in the EDCS is: “The

specifications in this International Standard were created by applying a set of guidelines. Specifications for proposed registered SRM concepts shall be created according to the following guidelines:..”

In particular, the new material in EDCS on referencing concepts should be adapted for use here.

UK_T082:

Throughout

Examples in this clause need to be put on proper style. Presently “For example,” is used.

UK_T083:

12.1, second sentence

The sentence “The membership of each set may extended through a process of registration.” is incorrect, as it seems to imply that the standard is “amended” by registration. Registered concepts are separate from those defined in the IS and are not “part of” the IS. Instead, it should say: “This International Standard allows new concepts to be specified by the registration of new concepts in each set.”

UK_T084:

12.1, second paragraph

The sentence “The following sets of SRM concepts may be extended by registration:” is again incorrect. Instead it should say “SRM concepts in the following sets may be registered:”

UK_T085:

12.7.2 and throughout the document

References to International Standards should be put into the proper format. For example, [1311] should instead be “ISO 31-1”.

Clause 13— Conformance

UK_T086:

13.3

Make the correction implied by the note: [Editors Note: The contents of the following table do NOT yet align with clause 11, API.].

UK_T087:

13.3.2, as well as elsewhere throughout the document

Numbering of each list within a subclause should re-start at “a” rather than continuing throughout a subclause.

Annex A—Mathematical foundations

UK_T088:

A.1, first sentence

The word “enumerates” reads oddly. Suggested rewording: “This annex identifies the concepts from mathematics assumed by this International Standard and specifies the notation used for those concepts.”

Annex B— Implementation notes

UK_T089:

B.1

This blank annex should be removed before the next version is published.

Annex C— UML diagrams for SRM concepts

UK_T090:

C.1, second, third and fourth sentences

For better wording and clarity, replace:

“SRM concepts are applicable to spatial objects of 1, 2, and 3 dimensions. For simplicity of presentation, this informative annex focuses only on the 3D case. Figure C. 1 illustrates the relationship between reference datum categories and (an extract of) standardized RDs.”

by

“While SRM concepts are applicable to spatial objects in 1, 2, and 3 dimensions, this informative annex presents only examples in 3D. Figure C. 1 specifies the relationship between reference datum categories and a subset of RDs.”

UK_T091:

Figures

Figure captions should be corrected to use proper nomenclature. For example, “Figure C. 1 - 3D Reference Datums” should be “Figure C. 1 - 3D RDs”.

UK_T092:

Figures

Are “Etc.” and ellipses (...) proper UML notation?

Annex D— RDs associated with celestial objects

UK_T093:

D1 Throughout (and referred to be UK G006)

Most RDs and other concepts related to non-Earth objects should be removed from the SRM. The exception is those concepts needed to support important solar and earth coordinate systems such as:

- 8.5.12 Celestiomagnetic SRF
- 8.5.13 Equatorial inertial SRF
- 8.5.14 Solar ecliptic SRF
- 8.5.15 Solar equatorial SRF
- 8.5.16 Solar magnetospheric SRF
- 8.5.17 Solar magnetic SRF
- 8.5.18 Heliospheric Aries ecliptic SRF
- 8.5.19 Heliospheric Earth ecliptic SRF
- 8.5.20 Heliospheric Earth equatorial SRF

Material to be removed includes, in particular, all data copied from the following (now obsolete) reference: Seidelmann, P.K., et al. Report of the IAU/IAG Working Group on Cartographic Coordinates and Rotational Elements of the Planets and Satellites: 2000 [online]. Celestial Mechanics and Dynamical Astronomy, vol. 82, p. 82-110, 2000. Dordrecht (Netherlands): Kluwer Academic Publishers, 2002. Available from World Wide Web: <<http://ipsapp002.lwwonline.com/content/search/4579/51/2/fulltext.pdf>>.

Note that the above document has been revised and will be re-published soon. A draft of the report was presented for discussion and completion at the IAU General Assembly in Sydney, Australia, in 2003. The fact that SC 24 is unaware of the revision activity says a lot about whether SC 24 has the necessary expertise to be establishing International Standards in this area.

(Note that the following 5 paragraphs duplicate UK G007, but are repeated here for completeness).

The agreed response to comment UK G003 on the SRM WD 7 (last version before CD) has not been implemented. That response stated:

“Response: There are two sets of concepts that are of concern. One is RDs for non-earth objects and the other is SRFs in 8.4.12 through 8.4.20. Establishing liaison with the appropriate international organizations and asking them to review the next version of this document solves the problems that this comment addresses. The next version should add planetodetic CS and planetodetic SRF.

There is a problem in dealing with parameters that are published by others and that may change over time. We need a unique way of designating a set of parameters (by code and label). We also need to be able to specify the epoch of validity of a parameter and to allow later parameters to supercede earlier ones.”

The appropriate international organizations have not been contacted and they did not review the SRM CD. Therefore, the offending material should be removed.

We know of no way that the issue in the second part of this response can be handled within an International Standard due to the long time frames that it takes to amend such a standard. Instead, the proper place for such parameters is in a document that can change more frequently, such as a register or a document controlled and revised by an organization (such as IAU/IAG Working Group on Cartographic Coordinates and Rotational Elements of the Planets and Satellites) with the technical expertise to update the parameter values.

Additional background for discussion:

The appropriate international organizations are:

1) The Joint IAU/IAG Working Group on Cartographic Coordinates

This working is charged with setting standards for the coordinate systems to be used in mapping the surfaces of bodies in the solar system. The standards resulting from their work are published periodically in the journal *Celestial Mechanics and Dynamical Astronomy*.

2) The IAU Working Group On The Celestial Reference

The WGICRS was created as a Working Group of IAU Division I as a continuation of similar groups since 1991 whose activities led to the adoption of the ICRF as a realization of the ICRS in 1997 and to an extensive set of resolutions in 2000 which provides the general framework of modern astrometry. The Working Group was renewed at IAU General Assembly 24 in August 2000 for a term of three years, with the goal of coordinating the work of astronomers to qualify, use, extend and promote the ICRS and prepare the recommendations relevant to these topics to be submitted to the IAU General Assembly in 2003. The Working Group comprises 39 members and is organized around six well identified tasks directed by a task leader. Each member of the working group has expressed personal interest in at least one of these task.

3) The International Society for Photogrammetry and Remote Sensing (ISPRS) WG IV/9:

Terms of Reference

- Status and technical definition of coordinate systems and geodetic control networks for mapping of planets and satellites
- Documentation of basic spacecraft datasets for extraterrestrial mapping, current and planned extraterrestrial mapmaking activities, and planetary cartographic products
- Development and documentation of new techniques for data acquisition and extraterrestrial mapping
- Development of GIS applications to support extraterrestrial exploration and science
- Web-based delivery of extraterrestrial map products and GIS data
- Cooperation with related working groups from ICA, IAU, NASA, and ESA

UK_T094:

Tables D.1 through D.5

RDs 147 and 148 are missing from this annex.

UK_T095:

D.3, Throughout

All TBDs should be removed before the next version is published.

Appendix E—ORMs

UK_T096:

Throughout

All TBDs should be removed before the next version is published.

UK_T097:

All tables

Code 194 seems to be missing.

UK_T098:

E.1, Throughout

Most RDs and other concepts related to non-Earth objects should be removed from the SRM. See UK_T093 on this subject for additional details.

UK_T099:**E.2, Throughout**

Should the comment to remove the non-Earth object ORMs from this annex not be accepted, then all TBDs should be removed before the next version is published.

UK_T100:**E.2.1, first sentence**

The words “(or registered)” should be removed. Registration is a form of specification, so saying “specified (or registered)” is redundant.

UK_T101:**E.2.2, first sentence and similar locations in other annexes**

Instead of “ORMs are specified by the contents of fields defined in Table 7.10” say “The fields of an ORM specification are defined in Table 7.10.”

UK_T102:**E.2.2, Table E.3**

The meaning of the grey fields needs to be explained.

UK_T103:**Table E.4**

There are two occurrences of ORM_RIKETS_1990. Either one should be deleted or the label should be distinguished between them.

UK_T104:**Table E.4**

The codes jump from 275 to 296 for no apparent reason and the codes between 275 and 296 are all missing. The codes should be renumbered.

Appendix F— Abbreviations and acronyms used in the construction of labels**UK_T105:****F**

A subclause structure should be added to better group text with the corresponding table. The subclause structure should reflect the titles of Tables f.2 and thereafter.

Appendix G— Change and deprecation plan**UK_T106:****G, Throughout**

This annex should be modified to track the accepted comments against the corresponding annex on the EDCS FCD ballot.

Annex H —Templates for registration proposals**Annex I —Conformance testing for spatial operations****UK_T107:****I, Throughout**

Implement the editor’s note in I.1

[Editors Note: Remove TRS/TRDS and move appropriate material to the conformance clause]

UK_T108:

I, Throughout

In its present form this annex adds little value. It should be re-written to be rigorous and of value or it should be removed in the next version.

Appendix J— Deprecated SRM concepts

UK_T109:

J, Throughout as well as Annex D and elsewhere, as appropriate

All numerical values taken from the following publications should be removed: [83502T] [HELM], and [DIGEST]. Instead either:

- a) a normative reference should be made to the latest version of these publications with a note on where the needed numbers are found in that specification; or
- b) the numerical values should be registered using the mechanisms established by ISO CD 19135 and cited in the SRM by normative reference.

In neither case should the numerical values themselves be included in the SRM itself. The reasons for not including such values in the SRM include:

- a) Those working on the SRM lack the expertise to create or independently verify the correctness of such values. All that we can do is copy the work of others and assume it is correct.
- b) The numerical values change over time and new editions of the cited references are published. Including such numerical values within an ISO standard will require that the standard be changed by amendment (a process taking at least 18 months).
- c) Users may be misled into thinking that the values in the SRM standard are definitive and take precedence over later revisions established by the owner of the above cited publications.
- d) The latest versions of the cited publications are the definitive references from which users should obtain the latest values.

To further support this comment, we note that the agreed response to comment UK G003 on the SRM WD 7 (last version before CD) has not been implemented. That response stated:

“Response: There are two sets of concepts that are of concern. One is RDs for non-earth objects and the other is SRFs in 8.4.12 through 8.4.20. Establishing liaison with the appropriate international organizations and asking them to review the next version of this document solves the problems that this comment addresses. The next version should add planetodetic CS and planetodetic SRF.”

There is a problem in dealing with parameters that are published by others and that may change over time. We need a unique way of designating a set of parameters (by code and label). We also need to be able to specify the epoch of validity of a parameter and to allow later parameters to supercede earlier ones.”

We know of no way that the issue in the second part of this response can be handled within an International Standard due to the long time frames that it takes to amend such a standard. Instead, the proper place for such parameters is in a document that can change more frequently, such as a register or a document controlled and revised by an organization (such as NIMA) with the technical expertise to update the parameter values.

Bibliography

UK_T110:

The reference USNOA should be supplied in the next version.

Editorial

Introduction

Clause 3— Normative references

Clause 3—Terms, definitions, symbols and abbreviated terms

Clause 4—Concepts

UK_E001:

4.5, EXAMPLE 2. Two lists that are labeled a, b, c

Change: Relabel the second set d, e, f.

UK_E002:

4.5, EXAMPLE 2, second line b. Missing word

Change: The constructed plane bound to the xz -plane reference datum contains.....

UK_E003:

4.5, EXAMPLE 3, second paragraph, second sentence. Grammatical errors

Change: The equatorial plane of the oblate spheroid determines the xy -plane, and its intersection with the oblate spheroid axis of rotation determines the origin point and the z -axis.

UK_E004:

4.5, EXAMPLE 3. Figure reference assumed to be incorrect

Change: (see ~~Figure 4.4~~ Figure 4.5)

Clause 5—Coordinate systems

UK_E005:

5.2.5.1

The first example is numbered “EXAMPLE 2.” Example numbering re-starts in each subclause. From 6.5.1 of the Directives, part 2:

“A single example in a clause or subclause shall be preceded by “EXAMPLE”, placed at the beginning of the first line of the text of the example. When several examples occur within the same clause or subclause, they shall be designated “EXAMPLE 1”, “EXAMPLE 2”, “EXAMPLE 3”, etc”.

UK_E006:

5.4, Table 5.5, CS type field

There is an extra space in “projection, 2D linear”.

Clause 6—Temporal coordinate systems

Clause 7—Reference datums and object reference models

Clause 9— Vertical offset models

Clause 10— Spatial operations

Clause 11—Application program interface

UK_E007:

11.2.6.1, last sentence

This sentence makes no sense. It should instead state something like: “The second set specifies the components that comprise a coordinate instance (that is, the value of a special position) defined with respect to an SRF.

UK_E008:

11.2.6.3.1

There are two occurrences of the word “Cartesian” being misspelled, once in the heading and once in the data type name. There is one case of it being miscapitalized. The proper spelling and capitalization is “Cartesian”. This should be checked throughout the entire standard.

UK_E009:

11.2.6.3.2

There are two occurrences of the word “Cartesian” being misspelled, once in the heading and once in the data type name. There is one case of it being miscapitalized. The proper spelling and capitalization is “Cartesian”.

UK_E010:

11.2.6.3.3, data type name

“Sperical” should be “Spherical”.

UK_E011:

11.2.6.3.3, data types of fields

The data types of the fields are misaligned.

UK_E012:

Table 11.16, Class

There is an extraneous space in the name of the class.

Annex A—Mathematical foundations

UK_E013:

A1, first sentence. Upper case letter needed

Change:International Standard.

Annex C UML diagrams for SRM concepts

UK_E014:

C.1, fifth sentence

“coponents” is misspelled.

UK_E015:

Throughout

Extra spaces should be removed.

UK_E016:

C, Throughout as well as elsewhere throughout the document

Callouts to figures are in all grey instead of normal underlined blue for hyperlinks. All should be presented in the same manner.

UK_E017:

C, Throughout

The notation should be defined and a reference to the ISO standard for UML given.

Annex D— RDs associated with celestial objects**UK_E018:****Table D.2**

The style (plain) used for labels and codes in Table D.2 is different from the font (bold) used for labels and codes in Table E.4 and elsewhere in Annex E. These should be made consistent.

Annex E—ORMs**UK_E019:****UK_G Table E.4**

Label ORM_GDZ8o should be bold.

Japan's comments on SC24 N 2472 Spatial Reference Model (CD)
2003-11-14, edited by Koreaki Fujimura

Japan_G001:

Global:

The CD draft has been prepared neglecting some dispositions of the previous Japan comments without any notice. Japan requests that such a discouraging and irritating action will not be repeated in the next review cycle.

Japan_T001:

Global (1, 4.5, 4.6.2, 4.9, etc.), time related matter:

It is not clear whether the time dependent (or dynamic) coordinate conversions are supported or not.

The scope clause should state whether the dynamic conversions are supported or to be supported in future extensions.

It should be explained in 4.6.2 that the temporal coordinate system will be used in defining dynamic features in 7.5 and 10.3.2.

It should be explained in 4.5 (and in 7.5 more definitely than the current text),

- what is the current (or future) usage of the celestial body dynamic binding categories in relation to other concepts,
- whether other dynamic bindings, e.g. artificial satellite movement or some hypothetical movements of celestial objects, will be supported in future or not.

It should be explained in 4.9 (and in 10.3.2 more definitely than the current text),

- how the dynamic ORM realization relates to other concepts, especially celestial body dynamic binding,
- what functionalities in the API clause correspond to the dynamic ORM realization.

Japan_T002:

4.1 whole text: The text here is a copy of 4.1.2 of the WD 7 with some trivial changes. The disposition of the previous Japan T015 comment on 4.1.2 of the WD 7

Response: Accepted in principle. Certain words ("coordinate systems") must be used more carefully. Examples might be given later and not in this overview, and care must be taken to give understandable examples

is totally neglected. The following is the minor modification of the previous comment.

There are some problems in this subclause:

- the first sentence informs nothing,
- the second sentence misleads readers that the SRM is designed only for "precise and unambiguous" criterion,
- it is not clear from the second paragraph what concepts are of main interest for users of this standard and what concepts are intermediate ones.

In order to increase the use of SRM, this subclause should be more friendly to new readers. Japan proposes an outline of alternative text as follows:

The purpose of the SRM is to provide an integrated system for describing spatial information (EXAMPLES: positions, directions, and distances) with the following features:

- most of the currently used coordinate systems are included,

- each coordinate systems are defined precisely and unambiguously in a uniform manner,
- coordinate conversions among those coordinate systems are supported.

To accomplish this, this International Standard takes the following approach:

- a. all the spatial information are given in the form of coordinate values (4.6 and Clause 5) based on some Spatial reference frame (SRF, 4.7 and Clause 8).
- b. Spatial Reference Frames are defined in terms of coordinate systems (4.6 and Clause 5) and Object reference models (ORM, 4.5 and Clause 7).
- c. description of the combining process of coordinate systems and ORMs is done using some Reference datums (RD, 4.4 and Clause 7).
- d. in order to describe the time-varying features, temporal coordinate system (4.6 and Clause 6) is introduced,
- e. in order to transform spatial information from one SRF to another SRF, spatial operations are discussed in Clause 10 and the Application programming interface is given in Clause 11.
- f. this International Standard also specifies common ideas that support the specification of all sets of SRM concepts including quality assurance, conformance, and registration (Clause 4, 12, 13).

In the rest of this clause, outlines of some important concepts are explained and concept specification principle is given.

Note: Japan cannot place Clause 9 in the above description (see Japan T008)

Japan_T003:

4.3, Figure .4.2:

The names and orientations of transformations “E1”, “E2”, and “T” should be shown in this figure.

Japan_T004:

4.5, Example 2:

The order of the second list

- a. oblate spheroid reference datum with major semi-axis ...
- b. z-axis reference datum, and
- c. xz-plane reference datum.

and the order of the first list

- a. The constructed directed line bound to the z-axis ...
- b. The constructed plane bound the xz-plane
- c. The major and minor semi-axes ...

should be aligned.

Japan_T005:

4.7, EXAMPLE 1:

The order of the list should be aligned with the order of the list in the definition of a spatial reference template described above.

Japan_T006:

4.7, after the last paragraph :

Add the forward reference to 8.7 where the concept of the SRF set is realized.

Japan_T007:

4.8, the first sentence:

The sentence

Real-world measurements of position may result in a 3rd-coordinate value referenced to a surface other than that of reference datums used in object reference models and spatial reference frames specified in this International Standard.

is hard to be understood by non-native English readers. Is the use of “result in” here common one? What phrase is matched with “and spatial reference frames ...”?

Japan_T008:

4.8:

The use of the vertical offset surfaces is still vague here because this subclause does not describe the relation to other concepts in spite of the disposition

Response: Accepted that Clause 4 should address this concept in greater detail, including its relationship to other concepts in the SRM.

of the previous Japan T037 comment

The relation between this clause and other clauses should be explained here (or in Clause 4).

The keyword "vertical offset" does not appear in Clause 5 to 8. Is it related to "ellipsoidal height" in 5.4.6?

Japan proposes to remove this subclause and Clause 9 at least from the main body (*1) because it is said in 9.4

In this International Standard the vertical coordinate for SRFs based on the SRFT_Celestiodetic and all map projection based SRF templates is defined as ellipsoidal height (see 5.3.6.1)

and it is of no use to specify the vertical offset specifications in 9.3 which will not be used anywhere.

*1) 4.8 and Clause 9 may be put into an informative annex as a warning not to mix up the vertical coordinates with elevation.

Japan_T009:

4.11:

Change: This subclause should be removed.

Rationale: Registration itself is not a concept specified in this document and there is no need to summarize Clause 12 here.

Japan_T010:

4.11:

(This comment becomes moot if the previous comment is accepted)

The first sentence misleads readers to think some new concepts in the same level as SRF may be registered. The wording in Clause 12 should be copied.

Japan_T011:

4.12:

Change: This subclause should be removed.

Rationale: Conformance itself is not a concept specified in this document and there is no need to summarize Clause 13 here.

Japan_T012:

5.2.2, the use of the term "replete":

The handling of "invented-here" term "replete" may put many people into unnecessary confusion. It should be explained before its first usage.

Change: Add a paragraph

Together with such common concepts, a newly introduced concept "replete" will be used. A set D is replete if all points in D belong to the closure of the interior of D . A replete set is a generalization of an open set that allows the inclusion of boundary points. Boundary points are important in the definitions of certain CSs.

before the sentence "This International Standard takes a functional approach to the construction of coordinate systems" in 5.2.1. Note 1 in 5.2.2 should be removed.

Rationale: The term "replete", which is used only twice, is very different from other terms, "one-

to one", "smooth", "smooth surface", "smooth curve orientation preserving", and "connected" in Note 1.

Japan_T013:

6.2.4 Coordinated universal time

Change:

1) The sentence

"Coordinated universal time (UTC) is the integrated temporal coordinate system that is used worldwide ..."

should be changed to

"Coordinated universal time (UTC) is not an integrated temporal coordinate described in 6.2.1 but a system identifying every time second of TAI that is used worldwide ..."

Rationale: As agreed (but neglected) in the disposition of the previous Japan T025 comment, UTC does not fulfill the definition of an integrated temporal coordinate system.

Japan_T014:

6.3, Table 6.2

It is misleading to have the title "Coordinate universal time" and the current text in the Description field.

If Japan Txxx is accepted, Table 6.4 should be removed because UTC is not an integrated temporal coordinate system to be specified.

Japan_T015:

6.3, (Table 6.2)

Add a parametric integrated temporal coordinate system (template) which enables to define a temporal coordinate system with a given (parametric) epoch and the unit of duration is SI second.

Change:

Table. 6.x -- Local UTC based coordinate system template

Field	Specification
Description	Local UTC based
Label	TCS_LOCAL_UTC
Code	2 (?)
Epoch	Given in a format defined by ISO 8601.
Unit of duration	SI second [I311]
Relationship to UTC	The epoch is given using in UTC but the time goes by independently of astronomical phenomena.
Reference type	?
References	?

Rationale: Such a temporal coordinate system, where

the ISO 8601 format should be allowed for the "Epoch" field (from the neglected disposition of previous Japan T027)

and the difference between TAI value is a constant, is necessary because this system saves the knowledge of the number of leap seconds since the start of 1958.

Japan_T016:

6.3:

TAI should be specified as a integrated temporal coordinate system for use.

Change: Use the current Table 6.2 as a template and the following changes should be applied:

- 1) "Label" should be TAI
- 2) "Relationship to UTC" should be "TAI - UTC varies from time to time, e.g., 10 seconds at the beginning of 1972 and 32 seconds at the beginning of 1999".

Rationale: Agreed (but neglected) in the disposition of the previous Japan T026 comment.

Japan_T017:

8.6, Table 8.30, the entry with Label "SRF_GEODETTIC_JAPAN_1991":

Change the label to ""SRF_GEODETTIC_KOREA_JAPAN_1991" or something like in order to clarify it's not an official Japan measuring system.

Japan_T018:

8.6, Table 8.30, the entry with Label "SRF_GEODETTIC_JAPAN_1991":

Okinawa, a part of Japan, should be removed from "Region".

Japan_T019:

8.6, Table 8.32:

Add an SRF set which consists of 19 plane coordinate systems and is in use in Japan officially if it will not delay the standardization.

Note: Japan discovered the SRF set facility very recently and might not prepare the material on this addition by the disposition meeting.

Japan_T020:

8.7, Table 8.37-8.51(esp. 8.42-8.46):

These tables should be merged into one table as given below.

Table 6.37 — GTRS indexing scheme and set members

Cell Code	Origin
$1 + 12 \cdot m + n$ ($m=0,1; n=0,\dots,11$)	$(-89, 5^\circ + m \cdot 1^\circ, -165^\circ + n \cdot 30^\circ)$
$25 + 24 \cdot m + n$ ($m=0,1; n=0,\dots,23$)	$(-87, 5^\circ + m \cdot 1^\circ, -172, 5^\circ + n \cdot 15^\circ)$
$73 + 36 \cdot m + n$ ($m=0,1; n=0,\dots,35$)	$(-85, 5^\circ + m \cdot 1^\circ, -175^\circ + n \cdot 10^\circ)$
$145 + 60 \cdot m + n$ ($m=0,\dots,3; n=0,\dots,59$)	$(-83, 5^\circ + m \cdot 1^\circ, -177^\circ + n \cdot 6^\circ)$
$385 + 72 \cdot m + n$ ($m=0,1; n=0,\dots,71$)	$(-79, 5^\circ + m \cdot 1^\circ, -177, 5^\circ + n \cdot 5^\circ)$
$529 + 120 \cdot m + n$ ($m=0,\dots,6; n=0,\dots,119$)	$(-77, 5^\circ + m \cdot 1^\circ, -178, 5^\circ + n \cdot 3^\circ)$
$1369 + 180 \cdot m + n$ ($m=0,\dots,10; n=0,\dots,179$)	$(-70, 5^\circ + m \cdot 1^\circ, -179^\circ + n \cdot 2^\circ)$
$3349 + 360 \cdot m + n$ ($m=0,\dots,119; n=0,\dots,359$)	$(-59, 5^\circ + m \cdot 1^\circ, -179, 5^\circ + n \cdot 1^\circ)$
$46549 + 180 \cdot m + n$ ($m=0,\dots,10; n=0,\dots,179$)	$(60, 5^\circ + m \cdot 1^\circ, -179^\circ + n \cdot 2^\circ)$
$48529 + 120 \cdot m + n$ ($m=0,\dots,6; n=0,\dots,119$)	$(71, 5^\circ + m \cdot 1^\circ, -178, 5^\circ + n \cdot 3^\circ)$
$49369 + 72 \cdot m + n$ ($m=0,1; n=0,\dots,71$)	$(78, 5^\circ + m \cdot 1^\circ, -177, 5^\circ + n \cdot 5^\circ)$
$49513 + 60 \cdot m + n$ ($m=0,\dots,3; n=0,\dots,59$)	$(80, 5^\circ + m \cdot 1^\circ, -177^\circ + n \cdot 6^\circ)$
$49752 + 36 \cdot m + n$ ($m=0,1; n=0,\dots,35$)	$(84, 5^\circ + m \cdot 1^\circ, -175^\circ + n \cdot 10^\circ)$
$49825 + 24 \cdot m + n$ ($m=0,1; n=0,\dots,23$)	$(86, 5^\circ + m \cdot 1^\circ, -172, 5^\circ + n \cdot 15^\circ)$
$49873 + 12 \cdot m + n$ ($m=0,1; n=0,\dots,11$)	$(88, 5^\circ + m \cdot 1^\circ, -165^\circ + n \cdot 30^\circ)$

Table 8.36 should be modified according to this change or it should be merged with a new Table 8.37.

Japan_T021:

11.2.5.4 and 11.2.7.2:

It is confusing (or illegal?) to use the same name "SRF" for a non-object data type and a class.

Japan_T022:

11.2.5.4:

All the SRF in Table 8.30 should be listed here in the same way as SRF_BRTISH_BATIONAL_GRID.

Japan_T023:

11.2.5.4:

All the SRF sets in Table 8.32 should be listed here in the same way as SRFS_ALABAMA_SPCS (now misspelled as SRF_... here and in 11.2.6.2.1).

Japan_T024:

11, Table 11.7:

Is it necessary to have the input parameter “source_srf” in the abstract operations of the class “SRF3DBase”, which may be interpreted as a source SRF?

Japan_T025:

11, Table 11.7:

The text in “Semantics” of the abstract operation “ChangeCoordinate3DSRF” of the class “SRF3DBase” should be changed to

This operation will change the 3D coordinates in this SRF to those in a target (specified) SRF. Source_coordinate must be within this SRF. The target_srf must be compatible with this SRF for conversion purposes

Japan_T026:

11, Table 11.7:

The parameter “target_srf” in the abstract operation “ChangeCoordinate3DSRF” and “ChangeDirectionSRF” of the class “SRF3DBase” should be moved from “Outputs” to “Inputs”.

Japan_T027:

11, Table 11.8:

The parameter “target_srf” in Abstract operation “ChangeCoordinate2DSRF” of the class “SRF2DBase” should be moved from “Outputs” to “Inputs”.

SEDRIS Organization Comments on Spatial Reference Model

ISO/IEC CD 18026

Submitted: 14 November 2003

GENERAL

SEDRIS_G001:

Operations Clause, Clause 7, Appendix E, and Appendix J

Change: Transformation parameters in the specification of an ORM should be used instead of “binding”. The term “embedding” or “embedding transformation” should not be used.

Rationale: Proper use of terms, and not overloading or overusing “embedding”.

Source: Meeting "G001"

SEDRIS_G002:

Through out

Change: Formulas are shown using mixed convention. Sometimes a constant equals an expression then in turn equals a function. In other cases it is reversed. Common convention should be followed (e.g. $F(\dots) = \dots = 1$)

Rationale: Consistency and convention

Source: Meeting "G002"

Clause 4

SEDRIS_G003:

All of Clause 4

Clause 4 does not make it clear why the various concepts presented in the clause are important to SRM or how they relate to each other. Statements that relate the individual concepts to each other are required. In addition, continuity sentences that bridge the various sub-topic to each other and to the overall SRM approach are missing. The sub-concepts in Clause 4 appear as independent pieces. A few specific comments that improve how the concepts relate are provided as technical and/or editorial SEDRIS comments, but a cohesive “editorial” revision of Clause 4 is still needed.

Rationale: Clause 4 has improved, but still lacks a continuity that ties all of the sub-concepts together and explains why these concepts are important to SRM, and how they are related to each other.

Source: Meeting "T005"

TECHNICAL

Clause 3 – Terms, definitions, symbols, and abbreviated terms

a SEDRIS_T001:

3.3.1 barycentre

Change: Remove.

Rationale: The term is not used.

Source: PB "T007"

SEDRIS_T002:

3.3.2. celestial sphere

Change: Remove.

Rationale: The term is not used.

Source: PB "T008"

SEDRIS_T003:

3.3.3. celestiocentric

Change: Remove.

Rationale: The term is not used in this sense. It is only used to name an SRFT.

Source: PB "T009"

SEDRIS_T004:

3.3.4. celestiodetic

Change: Remove.

Rationale: The term is not used in this sense. It is only used to name an SRFT.

Source: PB "T010"

SEDRIS_T005:

3.3.5. coordinate

Change: Remove.

Rationale: The term is defined differently in 5.2.2 (see footnote there).

Source: PB "T011"

SEDRIS_T006:

3.2 - Terms and definitions

Change: Add:

equatorial plane

plane through the centre of a planet that is normal to the rotational axis.

Rationale: Missing.

Source: PB "T012"

SEDRIS_T007:

Table 3.2 - Symbols

Change:

Symbo l	Definition
a	major semi axis length of an oblate spheroid
b	minor semi axis length of an oblate spheroid
h	celestial ellipsoidal height
k_0	map central scale factor
N	geoidal separation at a point
γ	convergence of the meridian
ϕ	celestial geodetic latitude
λ	celestial geodetic longitude
θ	spherical latitude or depression/elevation angle or cylindrical angle or (polar) angle
f	flattening (see Table 5.6)
\mathcal{E}	(first) eccentricity (see Table 5.6)
\mathcal{E}'	second eccentricity (see Table 5.6)
R_N	radius of curvature in the prime vertical (see Table 5.6)
R_M	radius of curvature in the meridian (see Table 5.6)
α	azimuth
ρ	radius or range
ξ	height
\mathbf{R}^n	vector space of n -tuples
h_e	elevation

Rationale: Incorrect or not used or missing

Source: PB "T013"

SEDRIS_T008:

Clause 3

Change: Add “spheroid” with the definition “a collective term for three-dimensional closed second order surfaces, including ellipsoids of revolution and tri-axial ellipsoids.”

Rationale: The collective term is not separately defined in the body of the text.

Source: Meeting "T013"

Clause 4 – Concepts

SEDRIS_T009:

4.1 Introduction

Change: Replace list with:

- a. object-space that is the space of a real or abstract object
- b. position-spaces that model object-space
- c. abstract coordinate systems that associate coordinates with points in position-space (see Clause 5);
- d. temporal coordinate systems that uses an abstract coordinate system to associate time with events (see Clause 6);
- e. position-space embeddings that identify points in position-space with points in object-space (see Clause 7);
- f. spatial coordinate systems that combine abstract coordinate systems with position-space embeddings to associate coordinates with points in object-space (see Clause 8);
- g. reference datums for defining reference points, curves, and surfaces used to bind position-space models to object-spaces (see Clause 7);
- h. object reference models that use reference datums to specify position-space embeddings (see Clause 7);
- i. spatial reference frames that bind abstract coordinate systems and object reference models in order to specify spatial coordinate systems (see Clause 8);
- j. vertical offset surfaces used to associate heights to positions(see Clause 9);
- k. spatial operations that relate positions, directions, and distances among different spatial reference frames (see Clause 10); and
- l. an abstract application program interface for spatial operations (see Clause 11).

Rationale: The list was based on earlier working drafts and is no longer accurate.
Source: PB "T014"

SEDRIS_T010:

4.1 Introduction
2nd paragraph.

Change: replace with:

To accomplish this, this International Standard specifies sets of abstract concepts that encompass the underling constructs that are implicit and/or explicit in a wide range of application domains and spatial information user disciplines. The specification of these abstractions promotes a clear set of terminology, and provides an extensible framework for both concept instances and operations. The key concepts that together comprise the SRM are:

Rationale: Explain the need for abstraction.
Source: PB " T000-2"

SEDRIS_T011:

4.1 Introduction

Change: Replace last paragraph with: "This International Standard also specifies concepts to support the SRM, including quality assurance, conformance, and registration."

Rationale: The word "idea" is an ill-defined term.
Source: Meeting "T001"

SEDRIS_T012:

4.2 Spatial objects and object space, 2nd sentence

Change:

~~Physical objects are real world objects.~~

Rationale: This sentence does not add information.
Source: PB "T015"

SEDRIS_T013:

4.2 Spatial objects and object-space, 2nd paragraph

Change: Replace the 1st sentence with: “Physical objects are divided into two sub-types: celestial objects and non-celestial objects.”

Replace the 2nd and 3rd sentences to remove “of interest to astronomers” and to add the word natural as follows: “Celestial object is further subdivided into planets, natural satellites, stars, and other astronomic objects.”

Rationale: Shorter, less controversial, and crisper wording

Source: Meeting "T002" and NGIT "E009"

SEDRIS_T014:

Figure 4.1

Change: Add ellipses “...” between “satellite” and “star” rectangles. Also, modify “Satellite” to “Natural Satellite”.

Rationale: To denote other possibilities and to match the new text.

Source: Meeting "T003"

SEDRIS_T015:

4.2 Spatial objects and object-space

Change: Replace third paragraph with: “The celestial object sub-types usually share similar physical properties, such as a generally spherical or ellipsoidal shape. These properties are used in defining reference datums and object reference models.”

Rationale: The phrase “some of the physical properties used to define some concepts in Clause 7” is vague. It is not clear what properties, or what concepts, are being referred to.

Source: Meeting "T004" and NGIT "E010"

SEDRIS_T016:

4.2 Spatial objects and object-space

Change: Paragraph before Example 1: “Object-space” should be italic and indexed.

Rationale: This is the definition of the term.

Source: Meeting "T006"

SEDRIS_T017:

4.2 Spatial objects and object-space

Change: Paragraph following Example 1: Delete first sentence. Move the 2nd sentence to end of the first paragraph of 4.2.

Rationale: Redundant. And wrong place.

Source: Meeting "T007"

SEDRIS_T018:

4.2 Spatial objects and object-space

Change: Paragraph 2 and 3 following Example 1: Move to the end of paragraph before Example 1.

Rationale: Correct flow.

Source: Meeting "T008"

SEDRIS_T019:

4.2 Spatial objects and object-space

Change: Add example for tower of Pisa. “Example 5 Tower of Pisa is a spatial object of the physical/non-celestial type.”

Rationale: Give example of a “fixed” non-celestial spatial object.

Source: Meeting "T009"

SEDRIS_T020:

4.3 Position space and normal embeddings

1st paragraph

Change: Title to “Position space and normal embedding models of object-space”.

And replace 1st paragraph with:

Position-space is an n-dimensional Euclidean space, for $n = 1, 2$, or 3 . Position-space serves as a vector space abstraction of object-space so that algebraic methods may be applied to spatial concepts.

A *normal embedding* of position-space is a distance preserving function from positions in position-space to points in object-space. Normal embeddings preserve important geometric properties. Position-space together with a normal embedding provides a specific algebraic model of object-space.

Algebraic models of object-space are not unique because there are infinitely many normal embeddings of an n-dimensional position-space. Figure 4.2 illustrates two distinct normal embedding models of an object-space.

Delete “Normal embeddings preserve important geometric properties.” from last paragraph. And change “Any two normal embeddings for the same object-space can be inter-converted with an affine transformation ~~called an embedding transformation.~~”

Rationale: Clarity.

Source: PB "T000-2"

SEDRIS_T021:

New. 4.4

Replace the first two paragraphs with:

A reference datum (RD) is a geometric construct in position-space that is used to specify an aspect of a normal embedding of position-space into object-space. Reference datums are defined for position-space, as points, directed curves, or oriented surfaces.

A reference datum is bound when the reference datum in position-space is identified with a corresponding constructed geometric entity in object-space. The term corresponding means that each (algebraic) reference datum is bound to a constructed spatial entity of the same geometric object type. That is: algebraic points are bound to spatial points, algebraic lines to spatial lines, algebraic curves to spatial curves, algebraic planes to spatial planes, and algebraic surfaces to spatial surfaces.

Example here.

Continue with “Figure 4.4 illustrates two distinct bindings ...”

Then:

This International Standard specifies a small set of reference datums for use in its own specifications, including points, axis lines, planes, and spheroids. Additional reference datums may be defined by registration, as described in 4.11.

Delete last paragraph.

Rationale: Clarity.
Source: PB "T000-2"

SEDRIS_T022:

4.4 Reference datums; formula in the Example

Change: Correct the formula.

Rationale: Incorrect value.
Source: Meeting "notes"

SEDRIS_T023:

4.5_Object reference models

Change: Add before first paragraph.

The process of specifying a specific normal embedding involves the (implicit or explicit) binding of RDs using application domain specific methodologies. These processes are outside of the scope of the SRM. However, the resulting specifications are abstracted and encapsulated in the notion of a object reference model. An object reference model identifies a set of RDs and constraints on their bindings, which guarantee a unique compatible normal embedding.

Rationale: Provides a needed transition from the previous subclause.
Source: PB "T000-2"

SEDRIS_T024:

4.5_Object reference models
EXAMPES 2, 3, 4

Change: A, B, C, D, E

A)
Remove Example 2[Ex. 3 becomes Ex. 2]

B)
Add Example [new] 3 between the 2nd and 3rd paragraphs following Figure 4.5 — An object reference model:

EXAMPLE 3 An object reference model template consisting of the following set of reference datums:

- a. oblate spheroid reference datum with major semi-axis *a* and minor semi-axis *b*,
- b. z-axis reference datum, and
- c. xz-plane reference datum,

and the following three binding constraints:

- a. The constructed directed line bound to the z-axis reference datum that contains the centre of the constructed oblate spheroid that is bound to the oblate spheroid reference datum.
- b. The constructed plane bound the xz-plane reference datum contains the constructed directed line bound to the z-axis reference datum.

- c. The major and minor semi-axes of the constructed oblate spheroid reference datum are a and b metres, respectively.

C) Reword Ex. 4:

EXAMPLE 4 4.5 Example 2 is a realization of the object reference model template in 4.5 Example 3.

D) EXAMPLE 5

... is a realization of the template in 4.5 ~~Example 2~~ Example 3 ...

E) EXAMPLE 6

... is a realization of the template in 4.5 ~~Example 2~~ Example 3 ...

Rationale: Example 2 was an ORM not an ORMT.

Source: PB "T000-2"

SEDRIS_T025:

Figure 4.10

Change figure labels

From "CS generating function" to "CS generating function"

From "3D orthonormal embedding" to "normal embedding"

Rationale: Correctness.

Source: PB "T000-2"

SEDRIS_T026:

4.6.3 Spatial coordinate systems

EXAMPLE 2:

Change: Replace with:

EXAMPLE 2 An engineering model is designed in abstract 3 dimensional Euclidean space using the Euclidean coordinate system. In this simple case, coordinate-space, position-space, and object-space are all identical (the generating function and normal embedding are both the identity operator), but each of the spaces is serving an implicit but distinct role.

Rationale: Clarity

Source: PB "T000-2"

SEDRIS_T027:

4.6.1 Abstract coordinate systems

Change: All occurrences (including Figure 4.7):

~~coordinate system~~ generating function

Rationale: Coordinate system function is not defined. Generating function is introduced in the 2nd paragraph.

Source: PB "T016"

SEDRIS_T028:

4.7 Spatial reference frames

Change: Revise paragraph after bullets to: "An object reference model determines a unique normal embedding of position-space. When that embedding is composed with the abstract coordinate system, the result is a spatial coordinate system for the object-space."

Rationale: Clarity

Source: Meeting "T011"

SEDRIS_T029:

4.8 Vertical offset surfaces, 2nd paragraph

Change:

Clause 9 specifies how the position of a point in the object-space of an object can be specified in part ~~as a distance from~~ with respect to a vertical offset surface.

Rationale: Vertical offsets are not necessarily a distance to the surface.

Source: PB "T017"

Clause 5 Coordinate systems

SEDRIS_T030:

All Clause 5 tables

Change: Verify whether Lambda* is correct and needed in all tables.

Rationale: Correctness.

Source: Meeting "notes"

SEDRIS_T031:

5.2.6 CS localization; 2nd and 3rd sentence in 2nd paragraph after Table 5.3.

Change: The order of "R3 in" needs to be changed. It should be "... in R3 ...". Same for 3rd sentence ("in R2")

Rationale: Correctness.

Source: Meeting "notes"

SEDRIS_T032:

5.3.3.4 Scale factor and point scale

and

5.3.3.5 Map scale

Change:

Remove 5.3.3.5 Map scale

And replace 5.3.3.4 Scale factor and point scale with:

One indicator of map projection distortion is the ratio of lengths between a line segment in coordinate-space and the corresponding curve in position-space. That ratio for an infinitesimally short line segment at a point is computed as the directional derivative of the map projection with respect to arc length and is called the *scale factor*. The scale factor along meridian at a point is denoted by j . The scale factor along a parallel at a point is denoted by k .

If a map projection is conformal, then the scale factors at a point are independent of direction ($j = k$). In this case the scale factor value at the point is called the *point scale factor* [HTDP]. Point scale functions for specific map projections are specified in Clause 10.

Scale factor varies over the area of a map projection. A nominal or representative scale value is sometimes used to estimate position-space distances based on coordinate-space distances.

Note: The factor used to compress an area of a map projection onto a printed map sheet is called a map scale. Map scale should not be confused with scale factor.

Rationale: Map scale is not correctly used elsewhere in the SRM.

The replacement is more precise and introduces the notation in Clause 10.

Source: PB " T000-2"

SEDRIS_T033:

5.3.4 Relationship to projection functions
paragraph before Figure 5.6

Change:

A cylindrical map projection is *tangent* if along the equator the ~~point~~ scale factor is equal to ~~the map scale~~ one. It is *secant* if the point scale is equal to the map scale along two parallels equally spaced from the equator in latitude. In that case the positive latitude is called the ~~standard latitude of origin. (see Error! Reference source not found.)~~.

Rationale: Map scale is incorrectly used. This definition of latitude of origin conflicts with last paragraph and the usage in the map projection specification tables.

Source: PB " T000-2"

SEDRIS_T034:

5.3.4 Relationship to projection functions
last paragraph

Change:

A conic map projection is *tangent* if along one parallel the ~~point~~ scale factor is equal to ~~the map scale~~ one. It is *secant* if the ~~point~~ scale factor is equal to ~~the map scale~~ one along two parallels in the same hemisphere. In that case the latitudes are called standard latitudes.

Rationale: Map scale is incorrectly used. This definition of latitude of origin conflicts with the usage in the map projection specification tables.

Source: PB " T000-2"

SEDRIS_T035:

5.3.5.1 False origin

Change:

The position (0,0) is called the *natural origin*. To avoid negative numbers in a region of interest in the coordinate-space of a map projection, it is common practice to add positive offsets to the values. The value added to the easting coordinate ~~#~~ component *u* is the *false easting*. The value added to the northing coordinate component *v* is the *false northing*. When either the false easting or false northing are non-zero ~~t~~The point with false coordinates (0,0) is the *false origin*.

Rationale: need to distinguish natural and false origins.

Source: PB " T000-2"

SEDRIS_T036:

5.3.5.2 Standard latitude, latitude of origin and central scale

Change: Replace with:

5.3.5.2 Longitude and latitude of origin and central scale

The longitude and latitude at the natural origin are called the *longitude of origin* and *latitude of origin* respectively. The scale factor in the direction of the parallel at the natural origin is the *central scale*. If the central scale of a map projection of sphere is equal to one, the map projection is a tangent map projection. If the central scale of a map projection of sphere is less than one, the map projection is a secant map projection. Specifying a central scale less than one is one generalization of secant map projections to the case of an oblate spheroid.

Rationale: The exiting use on map scale is incorrect. Standard latitude is not used as a CS parameter.

Source: PB " T000-2"

SEDRIS_T037:

Table 5.28 — Mercator

Table 5.29 — Oblique Mercator

Table 5.30 — Transverse Mercator

Table 5.31 — Lambert conformal conic

Table 5.32 — Polar stereographic

Table 5.33 — Equidistant cylindrical

b and

Table 5.5 — Coordinate system specification fields

Change:

Field	Specification
CS type	Surface (map projection) and 3D (augmented map projection).

Table 5.5 — Coordinate system specification fields

Field	Specification
CS type	One of: 3D linear, 3D curvilinear, surface linear, surface curvilinear, map projection, 2D linear curvilinear, 1D linear, or 1D curvilinear, or surface (map projection) and 3D (augmented map projection).

Rationale: Map projection is not a CS type as defined in 5.2.3

Source: PB "T018"

SEDRIS_T038:

5.4 CS specifications

Change:

Add

Table 5.x – Planetodetic 3D

Field	Specification
Description	Planetodetic 3D. Geodetic with longitude in opposite direction.
Label	CS_3D_PLANETODETIC
Code	
Function type	Generating function.
CS type	3D curvilinear.
Properties	Orthogonal.
CS parameters and constraints	a : major semi-axis length b : minor semi-axis length Constraints: $a > b$: (oblate spheroid) $a = b$: (spherical)
Coordinates	λ : planetodetic longitude in radians, ϕ : latitude in radians (θ in the spherical case), h : ellipsoidal height

Field	Specification
Domain of the generating function or mapping equations	$-\pi/2 < \phi < \pi/2$ $-\pi \leq \lambda < \pi$ $-b < h$
Generating function or mapping equations	$\mathbf{F}(\lambda, \phi, h) = \mathbf{F}_{\text{GD}}(-\lambda, \phi, h)$ <p>where:</p> \mathbf{F}_{GD} is the geodetic generating function.
Domain of the inverse of the generating function or mapping equations	$\{(x, y, z) \text{ in } \mathbf{R}^3 \mid (a - b) < x^2 + y^2 + z^2 \text{ and } 0 < x^2 + y^2\}$
Inverse of the generating function or mapping equations	$\mathbf{F}^{-1}(z, y, z) = \mathbf{F}_{\text{GD}}^{-1}(x, -y, z)$ <p>where:</p> $\mathbf{F}_{\text{GD}}^{-1}$ is the geodetic inverse generating function.
Figure(s)	See Table 5.15 Geodetic 3D
Notes	Similar to geodetic except that longitude is in the opposite direction. In particular, point on a planet surface rotate in the direction of increasing planetodetic longitude.
Reference type	IR
References	

Rationale: Missing.
Source: PB "T019"

SEDRIS_T039:

Clause 5, Table 5.6:

Second formula for first eccentricity is incorrect

Change: “ $\varepsilon = f^2 - fa$ ” to “ $\varepsilon^2 = 2f - f^2$ ”

Rationale: Correctness
Source: NIMA "T008a"

SEDRIS_T040:

Table 5.12 – Azimuthal spherical

Change: Verify equations and the diagram are correct.

Rationale: Correctness.
Source: Meeting “notes”

SEDRIS_T041:

Table 5.28 – Mercator

Change: The range for “latitude of origin” should be from $-\pi/2$ to $\pi/2$

Rationale: Latitude of origin should allow Southern hemisphere.
Source: Meeting “notes”

SEDRIS_T042:

Clause 5, Table 5.29:

Oblique Mercator formulas for the ellipsoidal case are given in Snyder.

Change: Add formulas for ellipsoidal oblique Mercator from Snyder. Consider reformulating as was done for transverse Mercator.

Rationale: Completeness

Source: NIMA "T008b"

SEDRIS_T043:

Table 5.29

Change: Oblique Mercator should become "Oblique Mercator Spherical" such that ellipsoidal case can be added. If ellipsoidal case material is ready in time, then it should be combined with Oblique Mercator. This also has impact on Clause 8 with SRFs.

Rationale: Correctness.

Source: Meeting "notes"

SEDRIS_T044:

Clause 5, Table 5.30:

Transverse Mercator formulas given are not valid at the poles. Specifically, the equation for isometric latitude (ψ) is not valid at latitude = $\pm 90^\circ$. The transverse Mercator is valid at the poles with a slight modification to the formulas.

Change: Formulas to be provided.

ACTION ITEM: Correct formulas to be provided in a separate document.

Rationale: Correctness

Source: NIMA "T008c"

SEDRIS_T045:

Clause 5, Table 5.30:

The domain of the generating function for the transverse Mercator does not include the fact that this projection is not defined for $\lambda - \lambda_{\text{origin}} = \pm \pi/2$ when $\varphi = 0^\circ$. This singularity should be noted.

Change: Add the following statement to the **Domain of the generating function or mapping equations** section of Table 5.30. "NOTE: Not defined at $\lambda - \lambda_{\text{origin}} = \pm \pi/2$ when $\varphi = 0^\circ$."

Rationale: Correctness

Source: NIMA "T008d"

SEDRIS_T046:

Clause 5, Table 5.30:

The formula for "z" incorrectly includes variables "a" and " k_0 "

Change: Remove these parameters from the equation for "z".

Rationale: Correctness, based on review of paper by L. P. Lee.

Source: NIMA "T008e"

SEDRIS_T047:

Clause 5 & 8

Table 5.31 — Lambert conformal conic

and

Table 8.26 — Lambert conformal conic SRFT

Change:

Remove parameters: false longitude, false latitude.

Rationale: Not used in formulae.

Source: PB "T000-2"

Clause 7

SEDRIS_T048:

7.2.2 Reference Datums

Change: Add table:

Table 7.x — 2D RDs of category point

RD label	Description	Position-space representation	RD code
RD_2D_ORIGIN	Origin	(0,0)	1
RD_2D_X_UNIT_POINT	x-axis unit point	(1,0)	2
RD_2D_Y_UNIT_POINT	y-axis unit point	(0,1)	3

Rationale: Missing.

Source: PB "T020"

SEDRIS_T049:

7.2.2 Reference Datums

Change: Add table:

Table 7.x — 2D RDs of directed curve

RD label	Description	Position-space representation	RD code
RD_2D_X_AXIS	x-axis	$F(t) = t(1,0)$	5
RD_2D_Y_AXIS	y-axis	$F(t) = t(0,1)$	6

Rationale: Missing.

Source: PB "T021"

SEDRIS_T050:

7.2.3 RDs of category ...; formula 7.1

Change: Correct the formula.

Rationale: Incorrect value.

Source: Meeting "notes"

SEDRIS_T051:

7.4.4 ORM Template, paragraph 5.

Change:

This International Standard specifies a set of minimal ORMTs for 2D and 3D position-space in Table 7.x and Table 7.9. The specification elements are defined in Table 7.8. Additional ORMTs may be registered in accordance with Clause 12.

And add table (or combine with Table 7.9):

Table 7.x — 2D ORMT specifications

ORMT label	ORMT specification	ORMT code
ORMT_2D_BI- AXIS_ORIGIN	<p>Description: x- and y-axes determined by directed perpendicular lines passing through the origin</p> <p>RDs:</p> <p>RD 1. RD_2D_ORIGIN</p> <p>RD 2. RD_2D_Y_AXIS</p> <p>RD 3. RD_2D_X_AXIS</p> <p>Binding constraints:</p> <p>BC 1. The constructed point bound to RD 1 shall be contained in the constructed directed line bound to RD 2 and the constructed directed line bound to RD 3.</p> <p>BC 2. The constructed directed lines bound the RD 2 and RD 3 shall be perpendicular.</p> <p>Notes:</p> <ol style="list-style-type: none"> The constructed point bound to RD 1 determines the origin of the embedding. The perpendicular directed lines passing through the origin uniquely determine the y-axis and x-axis of the embedding. 	

Rationale: Needed to support 3D SRFs.
Source: PB "T022"

SEDRIS_T052:
7.4.4 Example 3 b
and

Table 7.10 — ORM specification fields

and
NOTE 2, a.

Change:

b.a position (latitude 39° 13' 26,686" N) on the ellipsoid is identified to a position in the Earth object-space (Meades Ranch, Kansas, ~~United States~~ US),

and

Binding	<p>...</p> <p>If the ORM is an Earth-fixed ERM and if Greenwich, United Kingdom UK, is not contained in the x-positive half plane of the xz-plane of the embedding, then the</p> <p>...</p>
---------	--

and

NOTE 2

- a. Cases for which the xz-plane does not contain Greenwich, ~~United Kingdom~~ UK have relatively large ω_3 rotations.

Rationale: See WD7 UK comment UK_G002

Source: PB " T0000"

SEDRIS_T053:

7.4.5 Standardized ORM

Note 4.

Change:

Remove Note 4 and add the following paragraph.

Some ERMs that are specified in Table E.4 are based on local geodetic datums. Historic local geodetic datums often exhibit distortions with respect to the reference ORM. In those cases, the embedding transformation specified by the seven transformation parameters of the ORM is an approximation based on empirical measurements. In some of these cases, embedding transformation parameters for approximations based on different sets of measurements and/or sub-regions appear as separate entries in Table E.4. In those cases, the corresponding ORM labels share a common prefix derived from the name of the related local geodetic datum.

Rationale: Explains multiple ERMs with same description.

Source: PB " Tlast"

SEDRIS_T054:

Table 7.10 — ORM specification fields
and ANNEX J

Change:

Field	Definition
Binding	If object-fixed, either the designation "Reference ORM" or the values of the seven parameters $\Delta x, \Delta y, \Delta z, \omega_1, \omega_2, \omega_3$, and (if non-zero) Δs with respect to the object reference ORM.
Embedding parameters	If dynamic, a description of the binding. If the ...

and thought out **Annex E** and **Annex J**.

Rationale: "Binding" is an overloaded term in this clause and is confusing in this context. If fact 7.4.5 Standardized ORMs, 2nd paragraph, 1st sentence states:

"A standardized ORM does not include a specification of the binding." and yet one of the specification fields is labeled "Binding".

The parameters that appear here are defined in 7.3.3 last paragraph (after Note2) as "The *seven parameter embedding* specification".

Source: PB " T0000"

SEDRIS_T055:

7.5.1 Terminology and notation

Change:

Move last 3 paragraphs and Note 2 to Annex B
and rename:

7.5.1 Terminology

Rationale: The notation introduced in 7.5.1 no longer appears in 7.5, but is used in Annex B (see Annex B comments below)

Source: PB "T0000"

Clause 8

SEDRIS_T056:

8.1 Introduction, paragraph 1. Sentence 1.

Change: A full specification includes values for ~~the~~ CS parameters (if any) and a specification of the region of the space of the object.

Rationale: Not all CSs have parameters.

Source: PB "T023"

SEDRIS_T057:

8.2 Spatial coordinate systems

Change: Once an embedding of \mathbf{R}^m position-space into object-space is established, any ~~N-dimensional~~ abstract CS for \mathbf{R}^m ($m=1, 2, \text{ or } 3$) a region of that position-space specifies an ~~N-dimensional spatial CS~~ *spatial CS* that associates coordinates in coordinate-space to points in object-space. This association is a *binding of a CS via a spatial embedding*. This is illustrated in Figure 8.1 that illustrates a spatial surface CS bound with an embedding of \mathbf{R}^3 position-space to the 3D ~~space of an object-space~~. In this illustration, a surface coordinate (u, v) in coordinate-space is associated to a position (x, y, z) in the abstract position-space \mathbf{R}^3 . That position is then identified with a position in the space of an object via the embedding of \mathbf{R}^3 position-space determined by the selection of an origin and three unit points.

Rationale: Clarity and consistent terminology. The “dimension” a CS is not defined in Clause 5. CS type is defined. of a Definition of spatial CS should be in italics.

Source: PB "T024"

SEDRIS_T058:

8.3.2.1 ORM and CS compatibility, **end**

Change: As a further restriction, some ~~map projections~~ CSs are based on spheres only. These include:

e. ~~CS_MP_OBLIQUE_MERCATOR~~ has this restriction. , and

f. ~~CS_MP_EQUIDISTANT_CYLINDRICAL~~.

g. ~~_____~~

Rationale: CS_MP_EQUIDISTANT_CYLINDRICAL does not have this restriction. The constraint should not be limited (fro registration) to map projections.

Source: PB "T025"

SEDRIS_T059:

8.3.2.3 Coordinate names, Example, and throughout

Change: EXAMPLE 1 For a spherical CS, the assignment of SRF-specific names to the k^{th} -coordinates components of “right ascension” for λ , “declination” for θ , and “radius” for ρ .

Rationale: Correct terminology.

Source: PB "T026"

SEDRIS_T060:

8.3.2.4 Coordinate valid region, and throughout

Change: Replace all occurrences of “valid region” with “valid-region” [note hyphen added].

Rationale: In context such as “valid region description”, the term “valid” may be confused as adjective for “region description”

Source: PB "T027"

SEDRIS_T061:

8.3.2.4 Coordinate valid region, c.

Change:

- c. The SRF may be used in conjunction with other SRFs to form an atlas for a large region (See 8.7 SRF sets). In this case, the restrictions are used to control the pair-wise overlap of the spatial coverage of members of the SRF collection.

Rationale: Relevance.

Source: PB "T028"

SEDRIS_T062:

8.4 SRF induced surface reference frame

Change: The zero value 3rd-coordinate surface of the 3D CS of the SRFT_LOCAL_AZIMUTHAL_SPHERICAL_TANGENT_PLANE induces ~~the surface CS of the SRFT_SURFACE_LOCAL_AZIMUTHAL_TANGENT_PLANE~~ a lococentric surface azimuthal on the tangent plane of the SRF. ~~However, since that 3rd coordinate is not specified as a vertical coordinate, the SRFT_SURFACE_LOCAL_AZIMUTHAL_TANGENT_PLANE has a separate specification.~~ For the purpose of specifying an induced surface reference frame, the 3rd-coordinate θ : depression/elevation angle, is specified as a vertical coordinate.

[And corresponding changes to SRFT_LOCAL_AZIMUTHAL_SPHERICAL_TANGENT_PLANE and SRFT_SURFACE_LOCAL_AZIMUTHAL_TANGENT_PLANE]

Table 8.3 — SRFT directory

Surface local azimuthal tangent plane (<i>induced</i>)	SRFT_SURFACE_LOCAL_AZIMUTHAL_TANGENT_PLANE SRFT_LOCAL_AZIMUTHAL_SPHERICAL_TANGENT_PLANE
--	--

Remove: **8.5.8 Surface local azimuthal tangent plane SRF**

Rationale: Uniform treatment of induced surface reference frames simplifies the concept and the API.

Source: PB "T029"

SEDRIS_T063:

8.4 SRF induced surface reference frame

Change: Add note:

NOTE:Starting with a 3D SRF, SRM identifies surface SRFs on coordinate surfaces.Compare with the Geographic information – Spatial referencing by coordinates (DIS 19111) concept of compound coordinate reference frame which synthesizes a 3D reference frame from a surface and a vertical system. (See also 5.3.6.1 Augmentation with ellipsoidal height)

Rationale: Related 19111 concept.

Source: PB "T030"

SEDRIS_T064:

Table 8.2 — SRFT specification elements

Change:

CS coordinate names and/or symbols	SRF-specific names and/or symbols for the k^{th} -coordinate component names and/or symbols, or the phrase “The same as the CS.”, and a designation of the vertical coordinate if applicable.
---	---

Rationale: Completeness.

Source: PB "T031"

SEDRIS_T065:

Table 8.2 — SRFT specification elements

Change: Change “Object type” to “Object and object type”

Rationale: It includes both type and the object itself. Don’t want to use “instance”

Source: Meeting “notes”

SEDRIS_T066:

Table 8.3 — SRFT directory

Change:

CS Type		
Surface (map projection) and 3D (augmented map projection)		

Rationale: Closer correspondence to Table 5.7 — CS specification tables, (augmented) map projection are not “CS types”

Source: PB "T032"

SEDRIS_T067:

Table 8.3 — SRFT directory

Change:

CS type [From this order]	CS type [To this order]
3D	3D
Surface	Surface (map projection) and 3D (augmented map projection)
2D	Surface
Map projection and augmented map projection	2D

And the corresponding rows under each CS type.

Rationale: This arrangement has better dimensional continuity in object-space. In Table 5.7 -CS specification tables, Map projections are last because in that case the manner of gearing function specification differed from all other case. That criteria for ordering is not appropriate in clause 8.

Source: PB "T033"

SEDRIS_T068:

Table 8.4 — Celestiocentric SRFT

Change:

Specification element	Value
Description	Celestiocentric SRF. The generalization of geocentric spatial reference frames to include non-Earth objects.

Rationale: The name is not a description.

Source: PB "T034"

SEDRIS_T069:

8.5 SRF templates

Change: Add:

5.5.x Planetodetic SRF
Planetodetic SRFs shall be derived from the SRFT specified in Table 8.x

Table 8.x Planetodetic SRFT

Specification element	Value
Label	SRFT_PLANETODETIC
Code	
Description	Planetodetic SRF. Similar to Celestiodetic SRF with reversed sign for longitude.
Object type	Planet.
ORM constraint	Shall be derived from: ORMT_3D_OBLATE_SPHEROID or ORMT_3D_SPHERE
CS label	CS_3D_PLANTODETIC
CS coordinate names and/or symbols	The same as the CS. Ellipsoidal height is the vertical coordinate.
Template parameters	None.
CS parameter binding rules	CS parameters equal RD values: Oblate spheroid RD case - Major semi-axis a , inverse flattening f^{-1} Sphere RD case - Radius a
Coordinate valid region	No additional restrictions.
Notes	1. Used in planetary science.
Reference type	IR
References	TBD

Rationale: Missing

Source: PB "T035"

SEDRIS_T070:

8.5.6 Local tangent plane 3D SRF

Change:

Add:

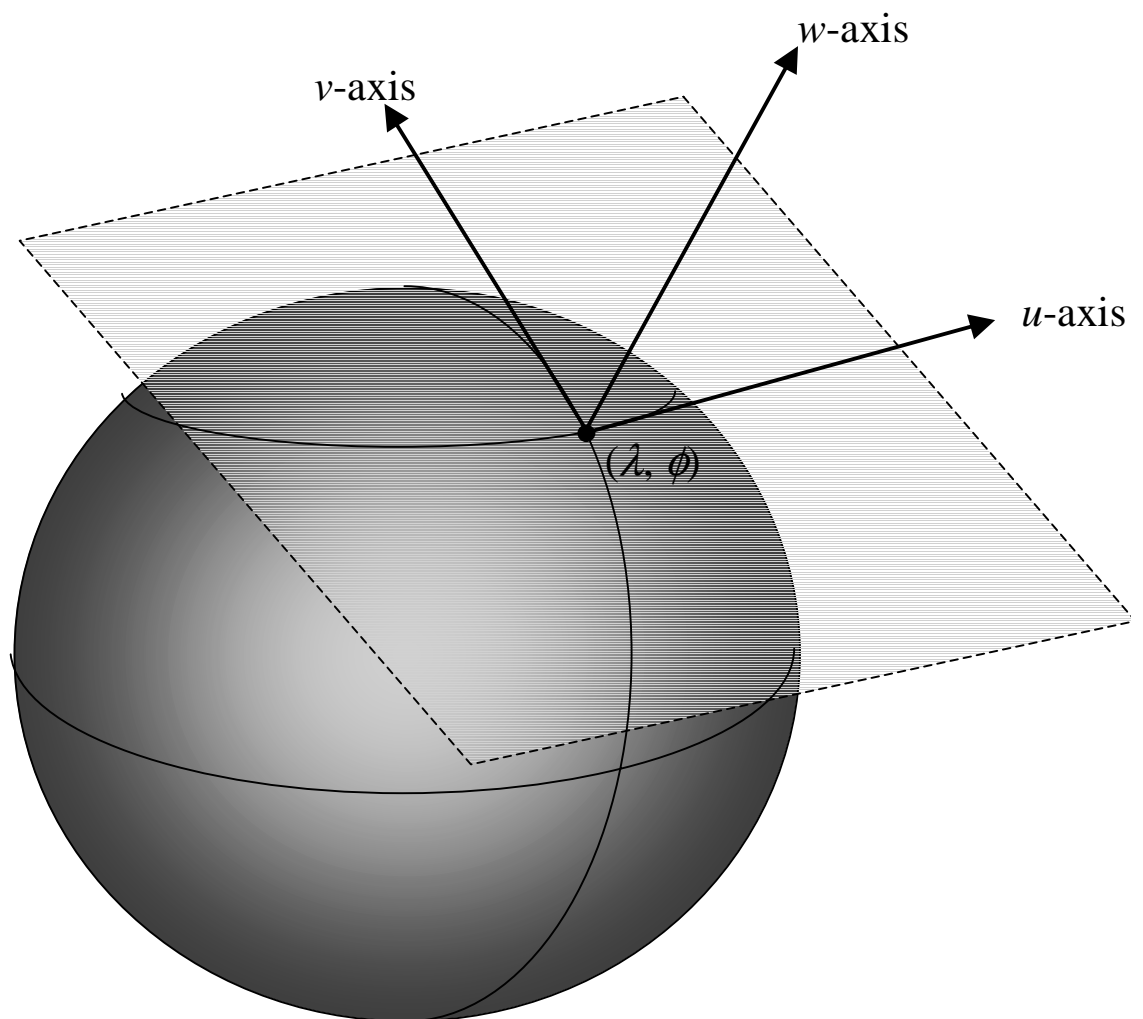
The case with parameters $\langle \alpha \rangle = 0$ and $h_{\text{sub}} = 0$ is illustrated in Figure 8.x

And add included figure. With Caption: Figure 8.x Local Tangent Euclidean SRF

Rationale: Clarity.

Source: PB “LTE”

Figure is here ->



SEDRIS_T071:

Table 8.5 — Local space rectangular 3D SRFT
and

Table 8.6 — Local space rectangular 2D SRF template

Change:

Specification element	Value
Label	SRFT_3D_LOCAL_SPACE_RECTANGULAR
Description	Local space rectangular 3D SRF. A 3D Euclidean reference frame for an abstract space.
Object type	2D Abstract object.
CS label	CS_32D_LOCOCENTRIC_EUCLIDEAN
Template parameters	Primary axis (1, 2, or 3). Direction of up (+1, -1). (up axis) Direction of forward (+1, -1). (forward axis)
CS parameter binding rules	$\mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \mathbf{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \text{ and } \mathbf{e}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix},$ <p>Delete:</p> $\delta(\text{direction}) = \begin{cases} 1 & \text{if direction is positive} \\ -1 & \text{if direction is negative} \end{cases}$ $\mathbf{E}(\text{axis}) = \begin{cases} \mathbf{e}_1 & \text{if axis is primary} \\ \mathbf{e}_2 & \text{if axis is secondary} \\ \mathbf{e}_3 & \text{if axis is tertiary} \end{cases}$ $\mathbf{u} = \delta(\text{up direction})\mathbf{E}(\text{up axis})$ $\mathbf{s} = \delta(\text{forward direction})\mathbf{E}(\text{forward axis})$ <p>Replace:</p> $\mathbf{E}(\text{axis}) = \begin{cases} +\mathbf{e}_1 & \text{positive primary axis} \\ +\mathbf{e}_2 & \text{positive secondary axis} \\ +\mathbf{e}_3 & \text{positive tertiary axis} \\ -\mathbf{e}_1 & \text{negative primary axis} \\ -\mathbf{e}_2 & \text{negative secondary axis} \\ -\mathbf{e}_3 & \text{negative tertiary axis} \end{cases}$ $\mathbf{u} = \mathbf{E}(\text{up axis})$ $\mathbf{s} = \mathbf{E}(\text{forward axis})$ $\mathbf{t} = \mathbf{s} \times \mathbf{u} \text{ (cross product)}$ $\mathbf{r} = \mathbf{0}$

And similarly in the 2D case.

Rationale: Simplification, delta not needed.
Source: PB "T036"

SEDRIS_T072:

Table 8.7 — Celestiodetic SRFT

Change:

Specification element	Value
Description	Celestiodetic SRF. The generalization of geodetic spatial reference frames to include non-Earth objects.

Rationale: The name is not a description.

Source: PB "T037"

SEDRIS_T073:

8.5.6. Local tangent plane 3D SRF

8.5.7 Local azimuthal spherical tangent plane SRF

8.5.10 Local cylindrical tangent plane SRF

and throughout.

Change: Change names to:

[3D SRFs]

Local tangent Euclidian 3D SRF

Local tangent azimuthal spherical SRF

Local tangent cylindrical SRF

[Surface SRFs]

Local tangent plane Euclidean 2D SRF

Local tangent plane azimuthal SRF

Local tangent plane polar SRF

Rationale: Local tangent plane 3D is non-descriptive. Use of “plane” in 3D SRF is confusing.

Source: PB "T038"

SEDRIS_T074:

Table 8.8 — Local tangent plane 3D SRFT

Change:

Specification element	Value
Description	Local Tangent Plane Euclidian 3D SRF. Euclidian 3D spatial CS with the zero 3 rd -coordinate surface tangent to the oblate spheroid RD and CS origin at the tangent point.

Rationale: The name is not a description.

Source: PB "T039"

SEDRIS_T075:

Table 8.9 — Local azimuthal spherical tangent plane SRFT

Change:

Specification element	Value
Description	Local tangent azimuthal spherical tangent plane SRF. Azimuthal spherical spatial CS with the zero 3 rd -coordinate surface tangent to the oblate spheroid RD and CS origin at the tangent point.
CS coordinate names and/or symbols	The same as the CS. θ . depression/elevation angle, is the vertical coordinate.

Rationale: The name is not a description. See SEDRIS_Txxx and SEDRIS_Txxx.
Source: PB "T040"

SEDRIS_T076:

Table 8.11 — Azimuthal 2D SRFT

Change:

c Specification element	d Value
e Description	f Azimuthal 2D SRF. An SRF for 2D abstract space based on the azimuthal CS.

Rationale: The name is not a description.
Source: PB "T041"

SEDRIS_T077:

Table 8.12 — Local cylindrical tangent plane SRFT

Change:

Specification element	Value
Description	Local tangent cylindrical tangent plane SRF. Cylindrical 3D spatial CS with the zero 3 rd -coordinate surface tangent to the oblate spheroid RD and CS origin at the tangent point.

Rationale: The name is not a description.
Source: PB "T042"

SEDRIS_T078:

Table 8.13 — Polar 2D SRFT

Change:

Specification element	Value
Description	Polar 2D SRF. An SRF for 2D abstract space based on the polar CS.

Rationale: The name is not a description.
Source: PB "T043"

SEDRIS_T079:

Table 8.14 — Celestiomagnetic SRFT

Change:

Specification element	Value
Description	Celestiomagnetic SRF. An spherical CS based SRF aligned with the magnetic dipole of a celestial object.

Specification element	Value
Object type	A rotating celestial object with a magnetic dipole axis distinct from its rotational axis.
ORM constraint	Shall be an celestiomagnetic dynamic binding category ORM from Table 7.17 , Table 7.20 .

Rationale: The name is not a description. Celestial object is too general for object type. (Editorial)Table links correctly but has wrong number.

Source: PB "T044"

SEDRIS_T080:

Table 8.15 — Equatorial inertial SRFT

Change:

Specification element	Value
Description	Equatorial Inertial SRF. A spherical CS based SRF aligned with the equator of a planet and the direction of the Sun at the vernal equinox (at a specified epoch).
ORM constraint	Shall be an equatorial inertial dynamic binding category ORM from Table 7.20 , Table 7.12 .
Notes	<ol style="list-style-type: none"> 1. See clause 7.5.2. 2. Star catalogues use right ascension and declination to specify directions.

Rationale: The name is not a description. (Editorial)Table links correctly but has wrong number.

Source: PB "T045"

SEDRIS_T081:

Table 8.16— Solar ecliptic SRFT

Change:

Specification element	Value
Description	Solar ecliptic SRF. A spherical CS based SRF aligned with the ecliptic plane of a planet and the direction of the Sun.
ORM constraint	Shall be an solar ecliptic dynamic binding category ORM from Table 7.14 .

Rationale: The name is not a description.

Source: PB "T046"

SEDRIS_T082:

Table 8.17— Solar equatorial SRFT

Change:

Specification element	Value
-----------------------	-------

Specification element	Value
Description	Solar equatorial SRF. A spherical CS based planet centred SRF aligned with the ecliptic plane and the rotational axis of the Sun.
ORM constraint	Shall be an solar equatorial dynamic binding category ORM from <u>Table 7.16.</u>

Rationale: The name is not a description.
Source: PB "T047"

SEDRIS_T083:
Table 8.18— Solar magnetospheric SRFT

Change:

Specification element	Value
Description	Solar magnetospheric SRF. A Euclidean 3D CS based planet centred SRF aligned with the direction to the Sun and the plane form with that direction and the magnetic dipole of the planet.
ORM constraint	Shall be an solar magnetic ecliptic dynamic binding category ORM from <u>Table 7.21.</u> <u>Table 7.22.</u>
CS label	CS_3D_SPHERICAL CS_3D_EUCLIDEAN
Notes	1. See clause 7.5.8. 2. In the case of planet Earth, it is also known as Geocentric solar magnetospheric SRF
References	TBD [Add to Bibliography: Russell, C. T., <i>Cosmic Electrodynamics</i> , 2, 1971, D. Reidel Pub. Co., Holland]

Rationale: The name is not a description. The SRF appears in the literature with Euclidean CS.
(Editorial)Table links correctly but has wrong number.
Source: PB "T048"

SEDRIS_T084:
Table 8.19— Solar magnetic SRFT

Change:

Specification element	Value
Description	Solar magnetic SRF. A Euclidean 3D CS based planet centred SRF with the z-axis aligned magnetic dipole and the xz-plane containing the Sun.
ORM constraint	Shall be an Solar magnetic dipole dynamic binding category ORM from <u>Table 7.24.</u>
CS label	CS_3D_SPHERICAL CS_3D_EUCLIDEAN

Rationale: The name is not a description. The SRF appears in the literature with Euclidean CS.
Source: PB "T049"

SEDRIS_T085:

Table 8.26 - Lambert conformal conic SRFT

Change:

Specification element	Value
Template parameters	Replace: ϕ_1, ϕ_2 : standard latitudes ($-\pi/2 < \phi_{\text{origin}} \leq \phi_1 \leq \phi_2 < \pi/2$) With: ϕ_1, ϕ_2 : standard latitudes ($-\pi/2 < \phi_1 \leq \phi_2 < \pi/2$)

Rationale: Extra constraint is not required in Table 5.31 - Lambert conformal conic CS

Source: PB "T050"

SEDRIS_T086:

8.7 SRF sets, b.

Change:

b. the valid regions of the set members ~~are~~ have non-overlapping interiors.

Rationale: Several SRFS do overlap at valid region boundaries.

Source: PB "T051"

SEDRIS_T087:

Table 8.29 — SRF specification elements

Change:

Specification element	Definition
Valid rRegion	Optional restriction of the domain of the CS to a valid region. If a valid region is specified, optionally an extended valid region. If both are unspecified, then there are no additional constraints on coordinate validity.

Rationale: Use defined terminology.

Source: PB "T052"

SEDRIS_T088:

Table 8.30 — SRFs, British National Grid, and throughout.

Change:

Specification element	Definition
Parameter values	Central meridian: $\lambda_0 = -2^\circ$ Scale on central meridian: $k_0 = 0,999\,601\,271\,7$ True origin: (49°N, 2°W). longitude of origin: $\lambda_0 = -2^\circ$ latitude of origin: $\phi_0 = +49^\circ$ central scale: $k_0 = 0,999\,601\,271\,7$ False easting: $u_F = -400\,000$ m. False northing: $v_F = 100\,000$ m.

And similarly where it occurs.

Rationale: Use the parameters and terminology defined in the SRFT table. In particular “True origin” appears often in the pages that follow, but it is not a defined term.

Source: PB "T053"

SEDRIS_T089:

Table 8.33 — SRF set specification elements

Change:

Specification element	Value
Region Coverage description	Optional restriction of the domain of the SRF set to a valid region. If a valid region is specified, optionally an extended valid region. If both are unspecified, then there are no additional constraints on coordinate validity. Optional description of the region corresponding to the union of the valid regions of all of the set members.
SRF set membership	A specification of the parameterization of the set members including valid region descriptions or specifications (If valid regions are specified, optionally extended valid region specifications), by listing or parameter algorithm. References to other specification tables may be used for this purpose (see Table 8.33).

Rationale:

(Region) It makes no sense for the valid region of the set to be larger than the union of the member valid regions, nor for it to be smaller or otherwise unrelated to the union.

(SRF set membership) Clarity and precision.

Source: PB "T054"

SEDRIS_T090:

Table 8.32 — SRFs

Change:

Specification element	Definition
Region Coverage description	Valid region description: State of Alabama (<u>US</u>).
etc.	etc.

Rationale: Valid region for a set is not defined.

Source: PB "T055"

SEDRIS_T091:

Table 8.32 — SRFs,

sub table SRFS_GTRS_GLOBAL_COORDINATE_SYSTEM

Change:

Specification element	Definition
-----------------------	------------

Specification element	Definition
Notes	<p>A set of 49 896 localized (but overlapping) SRFs, each with a valid region approximately 100 kilometres square, that are identified according to the geotile reference system indexing scheme.</p> <p>The commonly known GCS SRF set specifies the ORM_WGS_1984 as the ERM. The members of this SRF set are commonly known as cells. For much of the RD surface, each cell valid region covers one arc degree of geodetic latitude by one arc degree of geodetic longitude. However, near the poles, many arc degrees of longitude are grouped together into a single GCS cell since an arc degree of geodetic longitude becomes arbitrarily small near the poles. GCS cells are always one arc degree of geodetic latitude in extent. Within each GCS cell, a false origin offset is provided. The point of tangency is at the centre of the rectangular GCS cell, even if more than one arc degree of geodetic longitude falls within a GCS SRF cell. The SRFT_3D_LOCAL_TANGENT_PLANE azimuth parameter (α) is zero.</p>

Rationale: “(but overlapping) is confusing”.SRFT parameter specification does not belong in this table element
Source: PB "T056"

SEDRIS_T092:

Table 8.33 — SRF set member specification elements

Change:

Specification element	Definition
Valid rRegion	A valid region description or specification. Optionally an extended valid region description specification.

Rationale: Valid region description and valid region specification are separately defined terms that should not be confused.

Source: PB "T057"

Clause 9

SEDRIS_T093:

9.1 Introduction

Change:

Real-world measurements of position may result in a 3rd-coordinate value referenced to a surface other than that of ~~reference datums~~-RDs used in ~~object reference models~~ ORMs and ~~spatial reference frames~~ SRFs specified in this International Standard.

Such reference surfaces, termed vertical offset surfaces, may be specified in terms of ~~reference datums~~ the position-space embedding of an ORM. They are not, however, appropriate for direct use as reference datums. This International Standard specifies how the position of a point in ~~the object-space of an object~~ can be specified in part as a distance from a vertical offset surface.

Several SRFTs including SRFT_CELESTIODETTIC and all the map projection based SRFTs use ellipsoidal height as a 3rd coordinate component. This International Standard specifies how ellipsoidal height values are adjusted to approximate the distance of a position in object-space to the vertical offset surface.

Rationale: A VOS is not specified in terms of RDs.

This International Standard does not specify how a position of a point can be specified in part as a distance from a vertical offset surface. In general, the shortest line from a point to a VOS will not lie on a 3rd coordinate curve.

Source: PB "T058"

SEDRIS_T094:

9.2 - Object reference surfaces

Change:

g 9.2 - Object reference surfaces

9.2.1 - Model of a significant surface

An *object reference surface* (ORS) is a smooth surface defined with respect to (the embedding of) an ORM. An ORS specification includes both the smooth surface specification and the ORM specification upon which it is based. An ORS models an application-specific aspect of the object-space.

EXAMPLE — ~~An oriented surface RD component of an ORM is an ORS.~~

~~Two important cases of ORSs are reference spheroids (or spheres), and equipotential surfaces including geoids.~~

~~h 9.2.2 - Reference spheroids (and spheres)~~

[remove 9.2.2 entirely]

Rationale: The height measured from the spheroid of a ORM is just ellipsoidal height-- there are no offsets involved. 9.2.2 conflicts with concepts in clause 7.

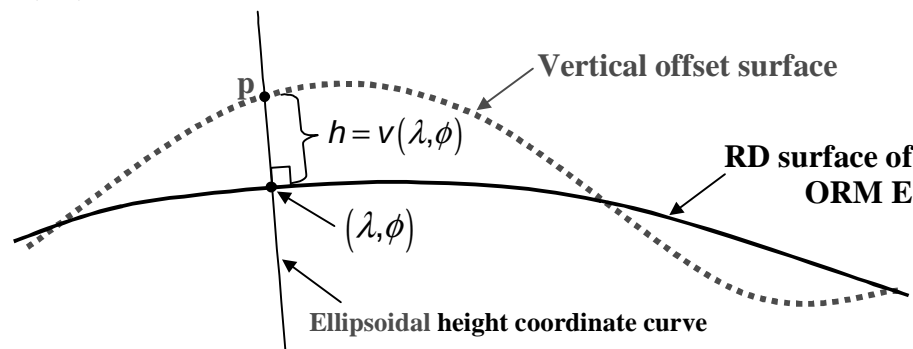
Source: PB "T059"

SEDRIS_T095:

9.2.4 Vertical offset surface

Change:

~~If Let S is be an ORS referenced to an ORM E, and if let R is be the SRF derived from SRFT_CELSTIODETIC with E, then the spheroidal height coordinate curve for each position p on the oblate spheroid (or sphere) RD surface of E may or may not intersect S. If the spheroidal ellipsoidal height coordinate curve at each position p on the ORS intersects S at p and no other one and only one point, then S is a vertical offset surface and the value of the ellipsoidal height coordinate component h at the intersect point is the offset surface separation at position p (see Figure 9.1). If (λ, ϕ) are the R surface geodetic longitude and latitude coordinate components for position p, then $v(\lambda, \phi)$ denotes the offset surface separation at position p.~~



~~If spheroidal ellipsoidal height coordinate curves intersect S uniquely in only a local region of the RD ORM then it is a local vertical offset surface, otherwise it is a global vertical offset surface.~~

Rationale: spheroidal height is not defined in this IS.

It is only necessary to check unique intersection where there exists some intersection.

Local regions are defined for ORMs not RDs.

Source: PB "T060"

SEDRIS_T096:

9.2.5 – Geoidal separation, last sentence.

Change:

The geoidal separation is usually specified as a table of values of $v(\lambda, \phi)$ ~~for a grid of oblate spheroid (or sphere)~~
~~RD positions~~ (λ, ϕ) an oblate spheroid ORM.

Rationale: Individual RDs do not determine embeddings.

Source: PB "T061"

SEDRIS_T097:

9.2.6 – Vertical coordinate value

Change:

Replace this sub clause with:

i 9.2.6 – Vertical offset height and elevation

In this International Standard the vertical coordinate for SRFs based on the SRFT_CELESTIODECTIC and all map projection based SRF templates is defined as ellipsoidal height (see 5.3.6.1).

Given a VOS with offset surface separation function $v(\lambda, \phi)$ for an ORM and a position \mathbf{p} , the *vertical offset height*, h_e is defined as follows:

Case 1 - The SRF is derived from SRFT_CELESTIODECTIC:

If (λ, ϕ, h) is the coordinate of \mathbf{p} , then $h_e = h - v(\lambda, \phi)$

Case 2 - The SRF is an augmented map projection based SRF:

If (u, v, h) is the coordinate of \mathbf{p} , then $h_e = h - v(\lambda, \phi)$

where (λ, ϕ) is determined by the inverse mapping equations:

$(\lambda, \phi) = (Q_1(u, v), Q_2(u, v))$.

If the VOS is a geoid, then h_e is termed elevation of \mathbf{p} with respect to the geoid. Note that (ellipsoidal height) - (elevation) = $v(\lambda, \phi)$.

Note 1: h_e approximates the distance from \mathbf{p} to the VOS. In general, the 3rd coordinate curve intersection with the VOS is not perpendicular to the VOS. When the intersection is not perpendicular, h_e does not equal the distance.

Note 2: VOS is similar in concept to vertical datum as defined in Geographic information – Spatial referencing by coordinates (DIS 19111). There, vertical datum is used to define a vertical coordinate reference system as part of a compound coordinate reference system.

Rationale: h_e should not be restricted to Celestiodectic SRFs.

Source: PB "T062"

SEDRIS_T098:

Table 9.2 Vertical offset surface specifications; Notes field of EGM96

Also in 3.3.

Change: Gravity to gravitational.

Rationale: Correctness

Source: Meeting "notes"

SEDRIS_T099:

9.4 Other vertical coordinate definitions

Change:

Move to follow 9.2.6

9.2.7 Other vertical coordinate definitions

And replace the first two paragraphs with:

In addition to vertical offset height (and elevation), different fields of application define other vertical coordinates, including:

Continue with:

orthometric height:

Rationale: The content of 9.4 is directly relevant to 9.2.6

Source: PB "T063"

SEDRIS_T100:

j 9.4 Other vertical coordinate definitions

Change: Add Note:

Note: Terrain elevation is not a standardized term and can have several different definitions and, consequently, different values depending on the application domain. The phrase "height above mean sea level" is ambiguous for several reasons. In addition to the varying definitions/approximations/models of the mean sea level VOS, height can be measured along a straight line normal to the surface or along a plumbline (plumbline curves may be empirical or depend on the gravity model used). These heights are called gravity-related heights in Geographic information – Spatial referencing by coordinates (DIS 19111). The Global Positioning System uses ellipsoidal height referenced to the WGS84 ellipsoid.

Rationale: [Editors note: Unaddressed from US_T049 comment on WD7]

Source: PB "T064"

Clause 10

SEDRIS_T101:

10.7.9.2 Point scale

Change to:

10.7.9.2 Scale factors

The equidistant cylindrical map projection is not conformal ~~so there are two scale factors. One of the scale factors in the direction of the u-axis is denoted by k while the scale factor perpendicular to the u-axis is denoted by j.~~

The scale factor in the meridional direction is denoted by *j*. The scale factor in the parallels direction is denoted by *k*.

Rationale: There are more than two scale factors and no point scale.

Source: PB "T000-2"

Clause 11

NOTE: The result of application of all SEDRIS comments on Clause 11 will produce a new and revised Clause 11. For convenience, this revised Clause 11 accompanies the comments from SEDRIS Organization as a separate and supplemental file. Reviewers may wish to refer to this file to see the proposed end results of applying all SEDRIS comments on Clause 11.

SEDRIS_T102:

Clause 11 (All)

Change: Re-order the sub-clauses as follows

- 11 Application program interface
 - 11.1 Introduction
 - 11.2 Non-object data types
 - 11.2.1 Overview
 - 11.2.2 Numbers
 - 11.2.3 Object_Reference
 - 11.2.4 Enumerated data types
 - 11.2.4.1 Introduction
 - 11.2.4.2 Axes_Direction
 - 11.2.4.3 Polar_Aspect (New-see comment below)
 - 11.2.5 Selection data types
 - 11.2.5.1 Introduction
 - 11.2.5.2 ORM
 - 11.2.5.3 Vertical_Offset_Surface
 - 11.2.5.4 SRFT, SRF, and SRFS and SRFS member codes (Revised-see comment below)
 - 11.2.5.5 Status_Code
 - 11.2.5.6 CSs
 - 11.2.5.6 RDs
 - 11.2.5.6 ORM templates
 - 11.2.6 Structured data types
 - 11.2.6.1 Introduction
 - 11.2.6.2 SRFT parameters
 - 11.2.6.2.2. LSR_3D_Parameters
 - 11.2.6.2.3. LSR_2D_Parameters
 - 11.2.6.2.4. LTP_Parameters
 - 11.2.6.2.5. ATP_Parameters
 - 11.2.6.2.6. Mercator_Parameters
 - 11.2.6.2.7. OM_Parameters
 - 11.2.6.2.8. LCC_Parameters
 - 11.2.6.2.9. PS_Parameters
 - 11.2.6.2.10. EC_Parameters
 - 11.3 Object types
 - 11.3.1 Introduction
 - 11.3.x Specification Format (New-see comment below)
 - 11.3.2 LifeCycleObject
 - 11.3.3 Private Objects
 - 11.3.3.1 Introduction
 - 11.3.3.3 Coordinate3D
 - 11.3.3.4 Coordinate2D
 - 11.3.3.5 CoordinateSurf
 - 11.3.3.6 Direction
 - 11.3.4 SRF Abstract classes
 - 11.3.4.1 Introduction
 - 11.3.4.3 BaseSRF
 - 11.3.4.4 BaseSRF3D abstract class
 - 11.3.4.5 BaseSRF2D abstract class
 - 11.3.4.6 SRFwithEllipsoidalHeightBase abstract class
 - 11.3.5 SRF concrete classes
 - 11.3.5.1 Celestiocentric SRF
 - (etc.)

- 11.3.5.25 Equidistant Cylindrical SRF
- 11.3.6 Object inheritance hierarchy (New-see comment below)
- 11.4 Standard SRFs
- 11.5 SRF Sets
- 11.6 Method precedence for Lifecycle objects (New-see comment below)

Rationale: Current order is inconsistent, has unnecessary forward references because of current order. Paragraph level is inconsistent.
Source: PB " T000"

SEDRIS_T103:

- 11.1 Introduction, 1st paragraph
- and
- 11.4

Change:

This International Standard specifies an API for the spatial operations in Clause 10. The API specifies non-object data types (see 11.2), and classes (object types) (see 11.3), ~~and non-object functions (see 11.4)~~ used to perform the spatial operations.

Rationale: There is no need for this standard to specify deg/radian conversions which are trivially understood (and remove 11.4).
Source: PB " T000"

SEDRIS_T104:

- 11.1 Introduction, 2nd & 3rd paragraphs

Change: Throughout: ~~operations~~ methods

Class is the term used to express the type of an *object*. Each class definition specifies the ~~operations~~ methods (if any) on the object. ~~Operations~~ Methods are specified by giving their syntax (input and output parameters), semantics (how the inputs interact with the state of the object and produce any outputs), and error conditions. In particular, the state of an object 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 active objects created as instances of a given class are reliably denoted by *object references*. Once created, objects exist and respond to method invocations ~~operations~~ until they are destroyed. The property of being created and existing until destruction is called the *object life cycle*. Objects inherit ~~interfaces~~ methods from other objects through the *subclass/superclass* relationship. Method inheritance is transitive: A subclass also inherits the method that have been inherited by its superclass.

Rationale:

1. Change though out this clause. Operations is a term used with a somewhat different meaning in clause 10 (where objects are not explicit). The term 'methods' is the standard term used in for the functionally associated with objects and would seem to be a better term than 'operations' which is much broader.
 2. Clarify how inputs interact with the state of the object.
- Source: PB " T000"

SEDRIS_T105:

- 11.1 Introduction, 5th & 6th paragraphs, and (name changes) throughout.

Change: Name changes:

The API specifies five classes that expose no operations: one SRF class; three coordinate classes: ~~3D_Coordinate~~, ~~2D_Coordinate~~, and ~~Surface_Coordinate~~ Coordinate3D, Coordinate2D, and CoordinateSurf; and one direction class: Direction (see 11.2.7.6). These classes are private types which hide all aspects of the ~~implementation~~ implementation of instances of these objects from the application.

The API specifies ~~four~~ seven abstract classes: Life_Cycle_Object, ~~SRF3Dbase~~, ~~SRFwithEllipsoidalHeightBase~~, and ~~SRF2Dbase~~ BaseSRF, BaseSRF3D, BaseSRFwithEllipsoidalHeight, BaseSRFwithTangentPlaneSurface, BaseMapProjection and BaseSRF2D (see 11.2). These abstract classes are used as the base classes from which subclasses including concrete classes inherit a common set of ~~operations~~ methods. Life_Cycle_Object includes the creation and destruction methods which all other classes inherit.

Rationale:

- Names that begin with digits (3D_Coordinate, etc) will have to be changed in most language bindings. Changing them here promote uniformity.
- (SRFwithEllipsoidalHeightBase) The juxtaposition of Height/Base can be confusing. Change the base type names to start with Base.
- (BaseSRFwithTangentPlaneSurface, BaseMapProjection) see comments below.

Source: PB " T000"

SEDRIS_T106:

11.2.4.2 Direction_Of_Forward

11.2.4.3 Direction_Of_Up

Change:

Direction_Of_Forward

~~This data type represents the direction of forward template parameter of SRFT_3D_LOCAL_SPACE_RECTANGULAR.~~

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

Direction_Of_Up

~~This data type represents the direction of up template parameter of SRFT_3D_LOCAL_SPACE_RECTANGULAR and SRFT_2D_LOCAL_SPACE_RECTANGULAR.~~

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

11.2.4.2. Axis_Direction

This data type represents the values of direction the parameter(s) of up SRFT_LOCAL_SPACE_RECTANGULAR_3D and SRFT_LOCAL_SPACE_RECTANGULAR_2D.

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

NEGATIVE_TERTIARY_AXIS)

Rationale: Combine Direction_Of_Forward and Direction_Of_Up. There is no need for two identical enumerations.

Source: PB " T000"

SEDRIS_T107:

11.2.4

Change: Add

11.2.4.3 Polar_Aspect

This data type represents the values of the polar aspect parameter of SRFT_POLAR_STERIOGRAPHIC.

Polar_Aspect ::= (ASPECT_NORTH,
ASPECT_SOUTH)

Rationale: Missing SRF parameter.

Source: PB " T000"

SEDRIS_T108:

11.2.4

Change: Add

11.2.4.4. Coordinate_Valid_Region

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

Coordinate_Valid_Region ::= (VALID, // contained in the valid region
EXTENDED_VALID, // contained in the extended valid region but not in the valid
region
DEFINED // contained in the CS domain but in either the valid or extended valid
regions)

Rationale: Missing SRF/Coordinate parameter (see comment below on changeCoordinateSRF).

Source: PB " T000"

SEDRIS_T109:

11.2.5.1 Introduction

Change: Add paragraph

A language binding may specify symbolic constants for individual selection data type values. In that case, the literal symbol shall be based on the label for that value (if any).

Rationale: Missing.

Source: PB " T000"

SEDRIS_T110:

11.2.5.2 ORM

Change:

11.2.5.2 ORM_code

The ORM_Code non-object selection data type specifies an ORM defined in Annex E.

ORM_Code ::= (< 1 : // implementation_dependent,

```

0      : INVALID // failure value for API methods that output an ORM_Code value.
1      : ORM_ACCRAABSTRACT
...    : // additional entries of the form C : L, for 2 <= C <= 489, where L is the label
denoting the concept also denoted by code C in Table E.3 through Table E.10.
490    : ORM_YACARE_URUGUAY
> 491  : // reserved for registration )

```

Rationale:

1. A code for an ORM should no be confused with an ORM.
2. INVALID needed for GetParameters method.
3. ORM_ACCA does not have code 1 (See annex E).

Source: PB " T000"

SEDRIS T111:

11.2.5.4 SRF

Change:

11.2.5.4 SRFT, SRF, and SRFS and SRFS member codes

The SRFT_Code, SRFS_Code, and SRF_Code non-object selection data types specifies an SRF Template, an SRF Set, or an SRF (see Clause 8).

```

SRFT_Code ::= (
    < 1      : // implementation dependent,
    1      : SRFT_CELESTIOCENTRIC,
    2      : SRFT_3D_LOCAL_SPACE_RECTANGULAR,
    3      : SRFT_2D_LOCAL_SPACE_RECTANGULAR,
    4      : SRFT_CELESTIODETTIC,
    5      : SRFT_3D_LOCAL_TANGENT_PLANE,
    6      : SRFT_LOCAL_AZIMUTHAL_SPHERICAL_TANGENT_PLANE,
    7      : SRFT_SURFACE_LOCAL_AZIMUTHAL_TANGENT_PLANE,
    8      : SRFT_2D_AZIMUTHAL,
    9      : SRFT_LOCAL_CYLINDRICAL_TANGENT_PLANE,
    10     : SRFT_2D_POLAR,
    11     : SRFT_CELESTIOMAGNETIC,
    12     : SRFT_EQUATORIAL_INERTIAL,
    13     : SRFT_SOLAR_ECLIPTIC,
    14     : SRFT_SOLAR_EQUITORIAL,
    15     : SRFT_SOLAR_MAGNETOSPHERIC,
    16     : SRFT_SOLAR_MAGNETIC,
    17     : SRFT_HELIOSPHERIC_ARIES_ECLIPTIC,
    18     : SRFT_HELIOSPHERIC_EARTH_ECLIPTIC,
    19     : SRFT_HELIOSPHERIC_EARTH_EQUATORIAL,
    20     : SRFT_MERCATOR,
    21     : SRFT_OBLIQUE_MERCATOR,
    22     : SRFT_TRANSVERSE_MERCATOR,
    23     : SRFT_LAMBERT_CONFORMAL_CONIC,
    24     : SRFT_POLAR_STEREOGRAPHIC,
    25     : SRFT_EQUIDISTANT_CYLINDRICAL,
    26     : SRF_BRITISH_NATIONAL_GRID,
    27     : SRF_ALABAMA_SPCS,
    28     : SRFS_UNIVERSAL_TRANSVERSE_MERCATOR,
    29     : SRFS_GTRS_GLOBAL_COORDINATE_SYSTEM
    > 30    : // reserved for registration )

```

```

SRFS_Code ::= (
    < 1      : // implementation dependent,
    0      : INVALID, // Failure value for API methods that output an SRFS_Code
              value.
    1      : SRFS_ALABAMA_SPCS,
    2      : SRFS_GTRS_GLOBAL_COORDINATE_SYSTEM
    etc. ...,

```

```

7 : SRF5_UNIVERSAL_TRANSVERSE_MERCATOR,
8 : SRF5_WISCONSIN_SPCS,
> 8 : // reserved for registration )

```

```

SRF_Code ::= (
    < 1 : // implementation dependent,
    0 : INVALID, // Failure value for API methods that output an SRF_Code
    value.
    1 : SRF_BRITISH_NATIONAL_GRID,
    2 : SRF_DELAWARE_SPCS,
    3 : SRF_GEOCENTRIC_EARTH_1984,
    4 : SRF_GEODETTIC_AUSTRALIA_1984,
    5 : SRF_GEODETTIC_AUSTRALIA_1990,
    etc. ...,
    14 : SRF_MARYLAND_SPCS,
    > 14 : // reserved for registration )

```

```

SRFS_Member_Code ::= (
    < 1 : // implementation dependent,
    0 : INVALID, // Failure value for API methods that output an
    SRFS_Member_Code value.
    >=1 : //To be interpreted in the scope of a given SRFS,

```

Rationale: 1. These 3 (SRFT, SRFS, SRF) need to be in separate code spaces as they are distinct concepts.
Source: PB " T000"

SEDRIS_T112:

11.2.5.4 Status_Code

Change:

The Status_Code non-object selection data type specifies the status codes returned by all operations 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 is further defined in the "Error conditions" field of each operation specification in Table 11.11 through Table 11.39.

```

Status_Code ::= (
    < 1 : // implementation_dependent,
    1 : SUCCESS, // the operation was performed successfully
    2 : INVALID_SRF,
    3 : INVALID_TARGET_SRF,
    4 : INVALID_SOURCE_COORDINATE,
    5 : EXTENDED_DESTINATION,
    6 : OPERATION_UNSUPPORTED,
    7 : INVALID_SOURCE_DIRECTION,
    8 : INVALID_INPUT,
    9 : CREATION_FAILURE,
    10 : DESTRUCTION_FAILURE,

```

Need to add:

```

    FLOATING_OVERFLOW,
    FLOATING_UNDERFLOW,
    FLOATING_POINT_ERROR,
    INVALID_SOURCE_SRF,
    INVALID_TARGET_COORDINATE,
    INVALID_CODE,
    >11 : // reserved for registration )

```

Rationale:

1. The abstract API should not specify the mechanism for the presentation of error conditions (see 11.3.1).
2. 5: EXTENDED_DESTINATION is not an error state or condition.
3. FLOATING_x needed in general for a computationally intensive API.
4. INVALID_SOURCE_SRF, INVALID_TARGET_COORDINATE, INVALID_CODE, required for error conditions (see comments below).

Source: PB " T000"

SEDRIS_T113:

11.2.6.2.1. Simple ORM Parameters

Change:

Remove.

Rationale: Do not need record type for single simple type.

Source: PB " T000"

SEDRIS_T114:

11.2.6.2.2. LSR_3D_Parameters

Change:

```
LSR_3D_Parameters ::= {      orm                ORM;
                             up_direction        Direction_Of_Up Axis_Direction;
                             forward_direction   Direction_Of_Forward Axis_Direction; }
```

Rationale: See comment on 11.2.4.1 above.

Source: PB " T000"

SEDRIS_T115:

11.2.6.2.3. LSR_2D_Parameters

Change:

```
LSR_2D_Parameters ::= {      orm                ORM;
                             forward_direction   Direction_Of_Forward Axis_Direction; }
```

Rationale: See comment on 11.2.4.1 above.

Source: PB " T000"

SEDRIS_T116:

11.2.6.2.5 ATP_Parameters

Change:

~~ATP~~ LTP_Parameters

Rationale: Why "A" in ATP? What does it stand for?

Source: PB " T000"

SEDRIS_T117:

11.2.6.2.5 OM_Parameters

Change:

11.2.6.2.5 OM_Oblique_Mercator_Parameters

This non-object data type specifies the parameters that correspond to SRFT_OBLIQUE_MERCATOR (see 0).

```

OM_Oblique_Mercator_Parameters ::= {      orm   ORM;
    longitude1      Long_Float;
    latitude1       Long_Float;
    longitude2      Long_Float;
    latitude2       Long_Float;
    central_scale   Long_Float;
    false_easting   Long_Float;
    false_northing  Long_Float;
    central_scale_factor Long_Float; }

```

Rationale:

1. The abstract specification should spell out names. Also LCC_, PS_, EC_, etc.
2. Missing or misnamed parameters (see clause 8).

Source: PB " T000"

SEDRIS_T118:

- 11.2.6.2.11. UTM_Parameters
- 11.2.6.2.12. GCS_Parameters

Change: Remove.

Rationale: These are SRF sets, not SRFTs. (See 11.2.5.4 comment)

Source: PB " T000"

SEDRIS_T119:

11.2.6.3 Coordinate Structures

CHANGE: Remove the coordinate structures from this clause since the API does not use them. Create a new clause to include structures that implementers of the API would use for storage mechanisms. Provide the following structures in that clause:

AZ_2D_Coordinate

This data type specifies the values of an azimuthal 2D coordinate.

```

AZ_2D_Coordinate ::= {
    azimuth      Long_Float;
    radius       Long_Float;
}

```

CD_Surface_Coordinate

This data type specifies the values of a Celestiodetic surface coordinate.

```

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

```

CD_3D_Coordinate

This data type specifies the values of a Celestiodetic 3D coordinate.

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

Euclidean_3D_Coordinate

This data type specifies the values of a 3D Euclidean coordinate.

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

Euclidean_2D_Coordinate

This data type specifies the values of a 2D Euclidean coordinate.

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

LTE_Surface_Coordinate

This data type specifies the values of a Local Tangent Euclidean surface coordinate.

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

LTE_3D_Coordinate

This data type specifies the values of a Local Tangent Euclidean 3D coordinate.

```
LTE_3D_Coordinate ::= {  
    u          Long_Float;  
    v          Long_Float;  
    w          Long_Float;  
}
```

LTA_Surface_Coordinate

This data type specifies the values of a Lococentric Tangent Azimuthal surface coordinate.

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

LTAS_3D_Coordinate

This data type specifies the values of a Lococentric Tangent Azimuthal Spherical 3D coordinate.

```
LTAS_3D_Coordinate ::= {  
    azimuth          Long_Float;  
    angle            Long_Float;  
    radius           Long_Float;  
}
```

LTC_Surface_Coordinate

This data type specifies the values of a Lococentric Tangent Cylindrical surface coordinate.

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

LTC_3D_Coordinate

This data type specifies the values of a Lococentric Tangent Cylindrical 3D coordinate.

```
LTC_3D_Coordinate ::= {  
    angle            Long_Float;  
    radius           Long_Float;  
    height           Long_Float;  
}
```

Polar_2D_Coordinate

This data type specifies the values of a polar 2D coordinate.

```
Polar_2D_Coordinate ::= {  
    angle            Long_Float;  
    radius           Long_Float;  
}
```

Spherical_3D_Coordinate

This data type specifies the values of an spherical 3D coordinate.

```
Spherical_3D_Coordinate ::= {  
    longitude        Long_Float;  
    latitude         Long_Float;  
    radius           Long_Float;  
}
```

EI_3D_Coordinate

This data type specifies the values of an equatorial inertial 3D coordinate.

```
EI_3D_Coordinate ::= {  
    right_ascension  Long_Float;
```

```

        declination      Long_Float;
        radius           Long_Float;
    }

```

Map_Projection_3D_Coordinate

This data type specifies the values of a map projection 3D coordinate.

```

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

```

Map_Projection_Surface_Coordinate

This data type specifies the values of a map projection coordinate.

```

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

```

RATIONALE: These data types are not required for the API, so it might confuse the user. They should be defined for users and implementers of the SRM since they provided defined structures that can be used such as being referenced from Part 1.

SEDRIS_T120:

11.2.7 Objects with no operations

11.2.7.1 Introduction

Change:

11.2.7 Private Objects ~~with no operations~~

11.2.7.1 Introduction

Private ~~Object~~ data types ~~with no operations~~ represent objects whose operations and attributes are ~~hidden~~ not exposed to the API user. Instances of these objects may be created and destroyed by operations on other objects specified in the API.

Therefore, their object references may be passed to and returned from operations. This allows an implementation of the API to store data that must be maintained for these object data types in whatever form is convenient for the implementation.

Rationale: As lifecycle objects, they in fact have methods/operations. Private Object is more descriptive.
Source: PB " T000"

SEDRIS_T121:

11.2.7.2 SRF

Change:

Move to 11.3.4 SRF Abstract classes
and change

11.2.7.2 BaseSRF

Table 11.2 — BaseSRF

Class	BaseSRF	
Description	An SRF (see 8.3.1).	
Superclass	LifeCycleObject	
Abstract method	Name	GetORMCode
	Semantics	Outputs the ORM_Code of this SRF.
	Inputs	None.
	Outputs	orm ORM_Code
	Error condition	No additional error conditions. (See 11.3.x a.)
Abstract method	Name	GetCodes
	Semantics	1. Outputs the SRFT_Code of this SRF3D instance. 2. If created by the CreateStandardSRF function, outputs a valid SRF_Code (otherwise 0). See 11.3.3.x 3. If created by an SRF set Class, outputs a valid SRFS_Code (otherwise 0) and a valid SRF_Member_Code (otherwise 0). See 11.3.3.y
	Inputs	None.
	Outputs	srft: SRFT_Code srf: SRF_Code srfs: SRFS_Code srfs_member: SRFS_Member_Code
	Error condition	No additional error conditions. (See 11.3.x a.)
Abstract method	Name	GetCSCode
	Semantics	Outputs the CS_Code of this SRF.
	Inputs	None.
	Outputs	orm CS_Code
	Error condition	No additional error conditions. (See 11.3.x a.)

Rationale: GetORMCode, GetCodes, GetCSCode are required to support data interchange using codes and are methods that (should be) common to all SRFs.

Source: PB " T000"

SEDRIS T122:

11.3 Object types

11.3.1 Introduction

Change:

11.3.1 Introduction

The API specified in this International Standard is presented in two groups: SRF objects and utility (non-object) functions. The SRF objects specifies all the operations specify methods that implement the spatial operations specified in Clause 10. The utility functions provide support for spatial operations.

The functionality of the API is provided using a class hierarchy with abstract classes that provides a means for providing common operations of subclasses for aid in specification. The abstract classes are not required to be implemented. The functionality is provided in the tables which provide the function method name, the semantics, inputs and outputs to the function, and the error conditions of the function. When the functionality is provided within the SRF class hierarchy, the functions are referred to as operation-methods. These operation-methods manipulate internal data (object state) and parameters passed in. The success condition is a nominal behaviour of all function methods and is not listed within the error conditions. The error conditions are listed in the descending order of priority. The second condition can only be true if the first condition is not present and so forth for the remaining error conditions.

EXAMPLE 1: In Table 11.2 BaseSRF, 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 Table 11.2 BaseSRF abstract method named GetORMCode “Outputs the ORM Code of this SRF”, and shows “Inputs: None”.

Language bindings may add additional error conditions and related binding specific mechanisms including memory allocation, the passing of inputs and outputs, and the presentation of operation status. Language bindings shall specify these mechanisms, since this part of 18026 does not restrict such mechanisms. Under an error condition, output values are undefined, but a language binding may specify the output values. When several error conditions apply to a method invocation, the first error condition detected by an implementation shall be presented as the operation status. A language binding mechanism for presentation of operation status shall support the association of a unique error Status_Code (11.2.5.5).

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 which returns the corresponding Status_Code would satisfy this requirement.

Rationale:

1. See comments above for use of “method” and elimination of “utility functions”. 2. The requirement “The error conditions are listed in the descending order of priority.” imposes unnecessary constraints on an implementer.

Source: PB " T000"

SEDRIS_T123:

11.3 Object types

Change: Add:

11.3.x Class Specification Format

Class types are specified in tables xx to xx with the following fields:

Field label	Field description
Class	Name of the private object class
Description	The corresponding SRM concept
Superclass	Specification of inherited functionality
(Abstract) method	Name
	Semantic
	Inputs
	Output
Error conditions	A list of Status_Code values. Each listed value specifies the condition for which it is applicable. Common error conditions (see below) are not listed in this field.

The method field of an Abstract class is labelled “Abstract method”. The method field of a concrete class is labelled “Method”. A subclass inherits all the methods of its superclass including methods that the superclass has inherited.

The method Error conditions field does not separately list the following common error conditions.

- 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 operations.
- FLOATING_OVERFLOW if a floating point overflow error occurred in the performance of the operation.
- FLOATING_UNDERFLOW if a floating point underflow error occurred in the performance of the operation.
- FLOATING_POINT_ERROR if floating point error (other than overflow or underflow) occurred in the performance of the operation.
- INVALID_INPUT if a Long_Float input is positive or negative infinity or a NAN value. If additional conditions apply, they are listed with this Status_Code in the (Abstract) Method Error conditions field.

Rationale:

Use format specification for consistency and compactness in presentation style.
Source: PB " T000"

SEDRIS_T124:

Table 11.7 — SRF3DBase

Change:

Table 11.7 — BaseSRF3DBase

Class	BaseSRF3DBase And throughout.
Description	This class is a 3D SRF. An abstract class representing the common elements of SRFTs that have a coordinate system of type 3D.
Superclass	LifeCycleObject BaseSRF

Rationale:

1. Name change (see comment above)
2. Improved description.
3. All SRF classes should inherit from BaseSRF.

Source: PB " T000"

SEDRIS_T125:

Table 11.7 — SRF3DBase (CreateCoordinate3D)

Change:

Table 11.7 — BaseSRF3DBase

Abstract operation method And throughout.	Name And throughout.	Create3Dcoordinate CreateCoordinate3D
	Semantics	This operation creates a 3D Coordinate3D for a specific this SRF from three ordered coordinate component values.
	Inputs	First_coordinate_component: Long_Float second_coordinate_component: Long_Float third_coordinate_component: Long_Float
	Outputs	coordinate: 3D Coordinate3D
	Error conditions	1. INVALID_SOURCE_COORD if the coordinate values are not in the domain of the coordinate system generating function as specified in 5.4.

Rationale: In accord with comments above.

Source: PB " T000"

SEDRIS_T126:

Table 11.7 — SRF3DBase (CreateDirection)

Change:

Table 11.7 — BaseSRF3DBase

Abstract operation	Name	CreateDirection
	Semantics	This operation accepts a 3D_Coordinate3D and the three direction components and a reference coordinate for a specific this SRF and creates a direction instances initialised with the values passed in. The direction vector is normalized.
	Input	source_srf: SRF source reference_coordinate: 3D_Coordinate3D first_direction_component: Long_Float second_direction_component: Long_Float third_direction_component: Long_Float
	Output	direction_out Direction
	Error conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. And throughout. 2. INVALID_SOURCE_COORD if source reference_coordinate is not in the CS domain of this SRF specified by source_srf or is invalid. 3. INVALID_SOURCE_DIRECTION if first_ direction_component, second_ direction_component, or third_ direction_component are invalid for source_srf . If the direction components are all zero.

Rationale:

1. In accord with comments above.
2. A direction vector is normalized by definition.
3. ~~source_srf: SRF~~And throughout. The invoking SRF is always an implicit input and should not be listed here.
4. ~~source~~reference_coordinate: ~~3D_Coordinate3D~~Use terminology consistent with 10.5.1.
5. ~~INVALID_SRF if source_srf was not successfully initialised through the API or is invalid.~~ See comment above (11.3.x Class Specification Format (a.))
6. If the direction components are all zero. This is the only error input.

Source: PB " T000"

SEDRIS_T127:

Table 11.7 — SRF3DBase (GetCoordinate3Dvalues)

Change:

Table 11.7 — BaseSRF3Dbase

Abstract operation	Name	GetCoordinate3Dvalues
	Semantics	This operation retrieves the three ordered coordinate components of a 3D coordinate instance previously initialised through the API
	Inputs	source_srf: SRF coordinate: 3D_Coordinate3D
	Outputs	first_coordinate_component: Long_Float second_coordinate_component: Long_Float third_coordinate_component: Long_Float
	Error conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. 2. INVALID_SOURCE_COORD if the coordinate was not a 3D coordinate, not initialised through the AP, or is not in the CS domain for this SRF. in source_srf. And throughout.

Rationale:

1. In accord with comments above.
 2. The phrase “not in” and SRF is not defined.
- Source: PB " T000"

SEDRIS_T128:

Table 11.7 — SRF3DBaseGetDirectionValues

Change:

Table 11.7 — BaseSRF3DBase

Abstract operation	GetDirectionValues	
	Semantic	This operation retrieves the reference coordinate and the three components of a direction instance previously initialised through the API.
	Input	source_srf: SRF direction_in Direction
	Output	reference_coordinate: Coordinate3D first_value Long_Float second_value Long_Float third_value Long_Float
	Error Conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. 2. INVALID_SOURCE_DIRECTION if the reference coordinate was not a 3D coordinate for this SRF or not initialised through the API.

Rationale:

1. In accord with comments above.
 2. reference_coordinate: This is an important value. Alternatively: Add a separate GetDirectionReferenceCoordinate() method.
- Source: PB " T000"

SEDRIS_T129:

Table 11.7 — SRF3DBaseChangeCoordinate3DSRF

Change:and add Note

Table 11.7 — BaseSRF3Dbase

Abstract operation	Name ChangeCoordinate3DSRF	
	Description Semantic	This operation will change the SRF of the passed in 3D coordinate to a specified SRF. Source_coordinate must be within the source_srf. The target_srf must be compatible with the source_srf for conversion purposes. This method changes the SRF representation of the spatial position specified by the input Coordinate3D, source_coordinate, from the source SRF, source_srf, to a Coordinate3D representation in this SRF in accordance with sub-clause 10.4.1. 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	source_srf: Base3DSRF source_coordinate 3D_Coordinate
	Outputs	target_srf SRF target_coordinate 3D_Coordinate region Coordinate_Valid_Region
	Error conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. 2. INVALID_SOURCE_DIRECTION_COORDINATE if source_coordinate is not in the CS domain of the SRF specified by source_srf. 3. INVALID_TARGETSOURCE_SRF , if target source_srf was not successfully initialised through the API or is invalid. 4. OPERATION_UNSUPPORTED , if source_coordinate cannot be changed to target_srf . If the source_srf is an SRF for a different spatial object. 5. INVALID_TARGET_COORDINATE if the spatial position is not in the CS domain of this SRF. 6. EXTENDED_DESTINATION , if the coordinate was changed to the target_srf this SRF, but it is now in the extended region of target_srf .

Note: The method ChangeCoordinate3DSRF requires an implementation to operate on a source coordinate from an unknown SRF object derived from BaseSRF3D. The clause 10 methodology for this operation supports this kind of extensibility. For example, if an implementation design requires each concrete (direct or indirect) subclass of BaseSRF3D to implement (private) methods for the generating function of the SRF and its inverse, and the transformation of the ORM to the reference ORM and its inverse, and if these methods are accessible (with a standard syntax) to all BaseSRF3D derived classes, then one generic way of changing the source coordinate from an otherwise unknown BaseSRF3D derived source SRF is to have the ChangeCoordinate3DSRF method of the target SRF: (1) apply the source SRF generating function to the source coordinate, (2) apply the source ORM transformation to the result, (3) then apply the target SRF inverse ORM transformation, and (4) finally apply the target SRF inverse generating function to produce the target coordinate. This computational scheme is presented for illustrative purpose only, and does not take into account error checking and various computational short cuts and efficiencies that an actual design would consider. (Some of these efficiencies are presented in Clause 10). This remark is also applicable to the ChangeDirectionSRF method.

Rationale:

1. In accord with comments above.
 2. Semantics Clarifies what operation is to be performed and reverses the source/target roles. Some languages can perform strong type checking on input types but not return values. Since the coordinate type of this SRF is known, but the SRF3Dbase subtype of the other SRF is unknown, this reversal of roles allows more flexibility in using LB capabilities without loss of functionality.
 3. ~~target_srf~~ — SRF This is not an output.
 4. region Coordinate_Valid_Region Needed to support valid regions.
 5. OPERATION_UNSUPPORTED Need to state why it is unsupported.
 6. INVALID_TARGET_COORDINATE The CS ranges for a valid pair of SRFs may not be equal sets.
- Source: PB " T000"

SEDRIS_T130:

Table 11.7 — SRF3Dbase ChangeDirectionSRF

Change:

Table 11.7 — BaseSRF3Dbase

Abstract operation	Name ChangeDirectionSRF	
	Semantics	<p>This operation will change the SRF of the passed in direction to a specified SRF. Source_direction must be within the source_srf. The target_srf must be compatible with the source_srf for conversion purposes</p> <p>This method changes the SRF representation of the direction by the input Direction, source_direction, from the source SRF, source_srf, to a Direction representation in this SRF in accordance with sub-clause 10.5.3.</p> <p>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.</p>
	Input	source_srf: Base3DSRF source_direction: Coordinate Direction Typo
	Output	target_srf SRF target_direction: Direction ref_coord_region Coordinate_Valid_Region
	Error conditions	<ol style="list-style-type: none"> INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. INVALID_SOURCE_DIRECTION if source_direction is not in the SRF specified by source_srf or is invalid. If the reference coordinate of the Direction is invalid or if the direction components are all zero. INVALID_TARGETSOURCE_SRF, if targetsource_srf was not successfully initialised through the API or is invalid. OPERATION_UNSUPPORTED, if the source_coordinate cannot be changed to target_srf. If the source_srf is an SRF for a different spatial object. INVALID_TARGET_COORDINATE if the reference location of the direction is not in the CS domain of this SRF. EXTENDED_DESTINATION, if the coordinates reference location of the direction was changed to the target_srf this SRF, but it is now in the extended region of target_srf this SRF.

Rationale:

- In accord with comments above.
- Semantics Clarifies what operation is to be performed and reverses the source/target roles. Some languages can perform strong type checking on input types but not return values. Since the coordinate type of this SRF is known, but the SRF3Dbase subtype of the other SRF is unknown, this reversal of roles allows more flexibility in using LB capabilities without loss of functionality.
- ~~target_srf SRF~~ This is not an output.
- region Coordinate_Valid_Region Needed to support valid regions.
- INVALID_SOURCE_DIRECTION Clarifies the error condition.
- OPERATION_UNSUPPORTED Need to state why it is unsupported.

7. INVALID_TARGET_COORDINATE The CS ranges for a valid pair of SRFs may not be equal sets and the condition should apply to Directions.

Source: PB " T000"

SEDRIS_T131:

Table 11.7 — SRF3DBaseEuclideanDistance

Change: Add method

Table 11.7 — BaseSRF3Dbase

Abstract method	Name	EuclideanDistance
	Semantics	Outputs the Euclidean distance (in metres) between the spatial points represented by Coordinate3Dsource_coordinate and target_coordinate.
	Inputs	source_coordinate Coordinate3D target_coordinate Coordinate3D
	Outputs	distance Long_Float
	Error conditions	1. INVALID_SOURCE_COORDINATE if source_coordinate is not in the CS domain of this SRF. 2. INVALID_TARGET_COORDINATE, if target_coordinate is not in the CS domain of this SRF.

Rationale: Clause 10 operation.

Source: PB " T000"

SEDRIS_T132:

Table 11.8 — SRF2DBase

Change:

Table 11.8 — BaseSRF2Dbase

Class	BaseSRF2Dbase
Description	This class is a 2D SRF. An abstract class representing the common elements of SRFTs that have a coordinate system of type 2D.
Superclass	LifeCycleObject BaseSRF

Rationale:

1. In accord with comments above.

2. Improved description.

Source: PB " T000"

SEDRIS_T133:

Table 11.8 — SRF2DBase

Change:

Table 11.8 — BaseSRF2Dbase

Abstract operation	Name GetCoordinate2Dvalues	
	Semantics	This operation retrieves the two ordered coordinate components of a 2D coordinate instance previously initialised through the API
	Inputs	source_srf: coordinate SRF 2D_Coordinate
	Outputs	first_value Long_Float second_value Long_Float
	Error conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. 2. INVALID_SOURCE_COORD if the coordinate was not a 2D coordinate, not initialised through the AP, or is not in source_srf the CS domain of this SRF.

Rationale: In accord with comments above.
Source: PB " T000"

SEDRIS_T134:
Table 11.8 — SRF2DBase

Change:

Table 11.8 — BaseSRF2DBase

Abstract operation	ChangeCoordinate2DSRF	
	Semantics	This operation will change the SRF of the passed in 2D coordinate to a specified SRF. Source_coordinate must be within the source_srf. The target_srf must be compatible with the source_srf for conversion purposes. This method changes the SRF representation of the spatial position specified by the input Coordinate2D, source_coordinate, from this SRF to a Coordinate2D representation in the target SRF, target_srf, in accordance with sub-clause 10.4.1.
	Inputs	source_srf: source_coordinate: SRF 2D_Coordinate
	Outputs	target_srf: SRF target_coordinate: 2D_Coordinate
	Error conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. 2. INVALID_SOURCE_COORDINATE if source_coordinate is not in the SRF specified by source_srf. this SRF. 3. INVALID_TARGET_SRF, if target_srf was not successfully initialised through the API or is invalid. 4. OPERATION_UNSUPPORTED, if source_coordinate cannot be changed to target_srf. 5. EXTENDED_DESTINATION, if the coordinate was changed to the target_srf, but it is now in the extended region of target_srf.

Rationale:

1. In accord with comments above.
 2. Semantics- Clarifies what operation is to be performed.
- Source: PB " T000"

SEDRIS_T135:

Table 11.8 — SRF2DBase

Change:Add method

Table 11.8 — BaseSRF2DBase

Abstract method	Name	EuclideanDistance
	Semantics	Outputs the Euclidean distance (in metres) between the spatial points represented by Coordinates2Dsource_coordinate and target_coordinate.
	Inputs	source_coordinate Coordinate2D target_coordinate Coordinate2D
	Outputs	distance Long_Float
	Error conditions	3. INVALID_SOURCE_COORDINATE if source_coordinate is not in the CS domain of this SRF. 4. INVALID_TARGET_COORDINATE, if target_coordinate is not in the CS domain of this SRF.

Rationale: Clause 10 operation

Source: PB " T000"

SEDRIS_T136:

Table 11.8 — SRF2DBase

Change:Remove method: ~~Free2Dcoordinate~~

Rationale: This lifecycle object inherits the destroy method..

Source: PB " T000"

SEDRIS_T137:

11.3

Change:Add new base class:

11.3.q BaseSRFwithTangentPlaneSurface

This is the base class for the LocalTangentPlane, LocalAzimuthalSphericalTangentPlane, and LocalCylindricalTangentPlane SRF classes. It is based on the SRF3DBase class. It adds methods for the surface CS that is induced on the third coordinate surface for zero which is a plane that is tangent to the oblate spheroid RD of the ORM of the SRF.

Class	BaseSRFwithTangentPlaneSurface
Description	This class is a 3D SRF for which the third coordinate surface for zero is a tangent plane to the oblate spheroid RD of the ORM of the SRF. A surface CS is induced on the third coordinate surface for zero.
Superclass	SRF3DBase

Class	BaseSRFwithTangentPlaneSurface	
Abstract operation	Name	CreateCoordinateSurf
	Semantics	Creates a surface coordinate on the tangent plane surface. The output is invalid if the operation does not succeed.
	Inputs	first_component Long_Float seond_component Long_Float
	Outputs	new_coordinate CoordinateSurf
	Error conditions	1. INVALID_SOURCE_COORDINATE if the input values are not in the domain of the induced surface CS generating function as specified in 5.4.
Abstract operation	Name	GetCoordinateSurfValues
	Semantics	Retrieves the component values of a surface coordinate on the tangent plane surface. The output is invalid if the operation does not succeed.
	Inputs	coordinate CoordinateSurf
	Outputs	first_component Long_Float seond_component Long_Float
	Error conditions	1. INVALID_SOURCE_COORDINATE if coordinate is not a valid surface coordinate in this SRF.
Abstract operation	Name	AssociateCoordinateSurf
	Semantics	Creates the CoordinateSurf 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).
	Inputs	coordinate_3D Coordinate3D
	Outputs	on_surface_coordinate: Surface_Coordinate
	Error conditions	1. INVALID_SOURCE_COORDINATE if coordinate_3D is not a valid 3D coordinate in this SRF.
Abstract operation	Name	PromoteSurfaceCoordinate
	Semantics	Creates a Coordinate3D representing the same location as specified by on_surface_coordinate (Promote surface coordinateto 3D coordinate).
	Inputs	surface_coordinate: CoordinateSurf
	Outputs	coordinate_on_the_surface: Coordinate3D
	Error conditions	1. INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate in this SRF.
Abstract method	Name	EuclideanDistance

Class	BaseSRFwithTangentPlaneSurface	
	Semantics	Outputs the Euclidean distance (in metres) between the spatial points represented by Surface_Coordinate, source_coordinate and target_coordinate.
	Inputs	source_coordinate CoordinateSurf target_coordinate CoordinateSurf
	Outputs	distance Long_Float
	Error conditions	1. INVALID_SOURCE_COORDINATE if source_coordinate is not in the CS domain of this SRF. 2. INVALID_TARGET_COORDINATE, if target_coordinate is not in the CS domain of this SRF.

Rationale: Adds needed methods common to 3 concrete classes.
Source: PB " T000"

SEDRIS_T138:

11.3.3 SRFwithEllipsoidalHeightBase abstract class, 1st paragraph

Change:

This is the base class for the celestiodetic SRF class and ~~all map projection SRF classes~~ BaseMapProjection Class. It is based on the SRF3DBase class. It adds ~~operations~~ methods for the surface CS that is induced on the zero ellipsoidal height surface. It also adds a ~~operation~~ method to create a local tangent plane SRF on a point on the zero ellipsoidal height surface.

Rationale: Clarity. See also comment below on BaseMapProjection
Source: PB " T000"

SEDRIS_T139:

Table 11.9 — SRFwithEllipsoidalHeightBase

Change:

Table 11.9 — BaseSRFwithEllipsoidalHeightBase

Class	BaseSRFwithEllipsoidalHeightBase
Semantic Description	This class is a 3D SRF with ellipsoidal height as third coordinate component. An abstract class representing the common elements of SRFTs that have an ORM with an oblate spheroid RD and a coordinate system of type 3D with ellipsoidal height (based on the RD) as the third coordinate component. A surface CS is induced on the third coordinate surface for zero.
Superclass	LifeCycleObject , SRF3DBase

Rationale:

1. Name change (see comment above)
 2. ~~Semantic~~ Description- Consistency
 2. Improved description.
 4. ~~LifeCycleObject~~- Inheritance is transitive. Do not introduce possible confusion with multiple inheritance.
- Source: PB " T000"

SEDRIS_T140:

Table 11.9 — SRFwithEllipsoidalHeightBase

Change:

Table 11.9 — BaseSRFwithEllipsoidalHeightBase

Abstract operation	CreateSurfaceCoordinate	
	Semantics	Creates a surface coordinate on the ORM surface. The output is invalid if the operation does not succeed.
	Inputs	source_srf: SRF first_component Long_Float seond_component Long_Float
	Outputs	new_coordinate Surface_Coordinate
	Error conditions	2. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. 3. INVALID_SOURCE_COORDINATE if the input values are not in the domain of the induced surface CS generating function as specified in 5.4.

Rationale: In accord with comments above.

Source: PB " T000"

SEDRIS_T141:

Table 11.9 — SRFwithEllipsoidalHeightBase

Change:

Table 11.9 — BaseSRFwithEllipsoidalHeightBase

Abstract operation	GetSurfaceCoordinateValues	
	Semantics	Retrieves the component values of a surface coordinate on the ORM surface. The output is invalid if the operation does not succeed.
	Inputs	source_srf: SRF coordinate Surface_Coordinate
	Outputs	first_component Long_Float seond_component Long_Float
	Error conditions	2. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. 3. INVALID_SOURCE_COORDINATE if coordinate is not a valid surface coordinate in the SRF specified by source_srf. this SRF.

Rationale: In accord with comments above.

Source: PB " T000"

SEDRIS_T142:

Table 11.9 — SRFwithEllipsoidalHeightBase

Change:

Table 11.9 — BaseSRFwithEllipsoidalHeightBase

Abstract operation	AssociateSurfaceCoordinate	
	Semantics	Creates the Surface_Coordinate associated with a 3D_Coordinate by setting the third coordinate component to zero and then converting that coordinate to its Surface CS representation (Truncate to surface).
	Inputs	source_srf: SRF coordinate_3D 3D_Coordinate
	Outputs	on_surface_coordinate: Surface_Coordinate
	Error conditions	2. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. 3. INVALID_SOURCE_COORDINATE if coordinate_3D is not a valid 3D coordinate in the SRF specified by source_srf. this SRF.

Rationale: In accord with comments above.

Source: PB " T000"

SEDRIS_T143:

Table 11.9 — SRFwithEllipsoidalHeightBase

Change:

Table 11.9 — BaseSRFwithEllipsoidalHeightBase

Abstract operation	PromoteSurfaceCoordinate	
	Semantics	Creates a 3D_Coordinate representing the same location as specified by on_surface_coordinate (Promote Surface to 3D).
	Inputs	source_srf: SRF surface_coordinate: Surface_Coordinate
	Outputs	coordinate_on_the_surface: 3D_Coordinate
	Error conditions	2. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. 3. INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate in the SRF specified by source_srf. this SRF.

Rationale: In accord with comments above.

Source: PB " T000"

SEDRIS_T144:

Table 11.9 — SRFwithEllipsoidalHeightBase

Change:

Table 11.9 — BaseSRFwithEllipsoidalHeightBase

Abstract operation	CreateLocalTangentPlaneSRF	
	Semantics	Creates a Local Tangent Plane SRF with natural origin at the input Surface_Coordinate. The created SRF has the same ORM as source_srf this SRF. The input surface_coordinate determines the tangent point geodetic parameters. The created SRF origin is determined from the remaining input parameters.
	Inputs	source_srf: _____ SRF surface_coordinate: Surface_Coordinate azimuth: Long_Float false_x_origin: Long_Float false_y_origin: Long_Float offset_height: Long_Float
	Outputs	new_local_tangent_plane_SRF: LocalTangenPlane 3DSRF
	Error conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. 2. INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate the SRF specified by source_srf. this SRF.

Rationale:

1. In accord with comments above.
 2. LocalTangenPlane 3DSRF - Strong typing where possible.
- Source: PB " T000"

SEDRIS_T145:

Table 11.9 — SRFwithEllipsoidalHeightBase

Change: Add method

Table 11.9 — BaseSRFwithEllipsoidalHeightBase

Abstract method	Name	VerticalSeparationOffset
	Semantics	Outputs the vertical separation offset (see 9.2.4) at the input surface coordinate between the oblate spheroid RD of this SRF and the specified VOS.
	Inputs	vos: VOS_Code, surface_coordinate: Surface_Coordinate
	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 vos is not a valid VOS_CODE 3. UNSUPPORTED_OPERATION if the VOS is not defined for the oblate spheroid RD of this SRF, or if the offset vertical separation is undefined at the surface location.

Rationale: Clause 9 operation

Source: PB " T000"

SEDRIS_T146:

Table 11.9 — SRFwithEllipsoidalHeightBase

Change:Add method

Table 11.9 — BaseSRFwithEllipsoidalHeightBase

Abstract method	Name	GeodesicDistance
	Semantics	Outputs the Geodesic distance (in metres) between source to target positions on the Surface of the oblate spheroid RD.
	Inputs	source_coordinate CoordinateSurf target_coordinate CoordinateSurf
	Outputs	distance Long_Float
	Error conditions	1. INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate this SRF. 2. INVALID_TARGET_COORDINATE if target_coordinate is not a valid surface coordinate for this SRF.

Rationale: Clause 10 operation

Source: PB " T000"

SEDRIS_T147:

Table 11.9 — SRFwithEllipsoidalHeightBase

Change:

Table 11.9 — BaseSRFwithEllipsoidalHeightBase

Abstract method	Name	EuclideanDistance
	Semantics	Outputs the Euclidean distance (in metres) between the spatial points represented by Surface_Coordinate source_coordinate and target_coordinate.
	Inputs	source_coordinate CoordinateSurf target_coordinate CoordinateSurf
	Outputs	distance Long_Float
	Error conditions	3. INVALID_SOURCE_COORDINATE if source_coordinate is not in the CS domain of this SRF. INVALID_TARGET_COORDINATE, if target_coordinate is not in the CS domain of this SRF.

Rationale: Clause 10 operation

Source: PB " T000"

SEDRIS_T148:

k 11.3.3.w BaseMapProjection Class

Change: Add

11.3.3.w BaseMapProjection Class

Class	BaseMapProjection	
Description	An abstract class representing the common elements of SRFs that have a map projections coordinate system of type 3D.	
Superclass	BaseSRFwithEllipsoidalHeight	
Abstract method	Name	ConvergenceOfTheMeridian
	Semantics	Outputs the Convergence of the Meridian in radians at a position on the Surface of the oblate spheroid RD.
	Inputs	surface_coordinate CoordinateSurf;
	Outputs	gamma Long_Float
	Error conditions	1. INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate this SRF.
Abstract method	Name	PointScale
	Semantics	Outputs the point scale at a position on the Surface of the oblate spheroid RD.
	Inputs	surface_coordinate Surface_Coordinate;
	Outputs	scale Long_Float
	Error conditions	1. INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate this SRF.
Abstract method	Name	MapAzimuth
	Semantics	Outputs the map azimuth in radians at from a source to target positions on the Surface of the oblate spheroid RD.
	Inputs	source_coordinate CoordinateSurf; target_coordinate CoordinateSurf;
	Outputs	azimuth Long_Float
	Error conditions	1. INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate this SRF. 2. INVALID_TARGET_SRF, if target_srf is not a valid surface coordinate in this SRF.

Rationale: Supplies missing map projection operations.

Source: PB " T000"

SEDRIS_T149:

11.3.3.1 LifeCycleObject

Change:

Move this sub-clause to precede 11.2.7.

Rationale: Define objects before using them.

Source: PB " T000"

SEDRIS_T150:

Table 11.10 — LifeCycleObject

Change:

Table 11.10 — LifeCycleObject

Class	LifeCycleObject
Description	This is the most basic abstract class from which all other classes inherit. An object derived from LifeCycleObject is invalid until the Create method is successfully invoked. A valid object is invalid after the Destroy method is invoked.
Superclass	None

Rationale: Improve description.

Source: PB " T000"

SEDRIS_T151:

Table 11.10 — LifeCycleObject

Change:

Table 11.10 — LifeCycleObject

Abstract operation	Name Create	
	Semantics	This operation creates an instance of an object. An implementation may perform memory allocation and/or object initialization as part of this method.
	Inputs	None Specific inputs are specified in concrete classes which are directly or indirectly sub-classed from this class type.
	Outputs	objRef: Object_Reference
	Error conditions	1. CREATION_FAILED if the object instance could not be created.

Rationale:

1. In accord with comments above.
2. Improve semantic description.
3. Some subclasses have inputs for the Create method.

Source: PB " T000"

SEDRIS_T152:

Table 11.10 — LifeCycleObject

Change:

Table 11.10 — LifeCycleObject

Abstract operation	Name Destroy	
	Semantics	This operation destroys an instance of an object and releases any storage used by the object instance. An implementation may perform memory deallocation and/or object finalization as part of this method.
	Inputs	objRef: Object_Reference
	Outputs	None.
	Error conditions	1. DESTRUCTION_FAILED_FAILURE Typo. if the object instance could not be destroyed. if the object was not successfully created through the API.

Rationale:

1. In accord with comments above.
 2. Improve semantic description.
 3. Explain why it could not be destroyed.
- Source: PB " T000"

SEDRIS_T153:

11.3.3.2 SRF concrete classes

Change:

1. This should be at the same level as Abstract class (11.3.4).
2. It may be helpful to break this clause into sub-clauses based on superclass: BaseSRF3D, BaseSRFwithEllipsoidalHeight, BaseSRFwithTangentPlane Surface, BaseMapProjection, and BaseSRF2D

Rationale: Organization

Source: PB " T000"

SEDRIS_T154:

11.3.3.2 SRF concrete classes, 1st paragraph

Change:

SRFs are ~~specified with~~ represented as instances of concrete classes in the tables below.

Rationale: SRFs are specified in clause 8 not here.

Source: PB " T000"

SEDRIS_T155:

Table 11.11 — Celestiocentric

Change:
Table 11.11 — Celestiocentric

Class	Celestiocentric
Description	An instance of this class corresponds to an instance of SRFT_CELESTIOCENTRIC
Superclass	LifeCycleObject , SRF3Dbase

Class	Celestiocentric	
Operation	Name Create	
	Semantics	Overrides the Create operation on the superclass LifeCycleObject. Creates a celestiocentric SRF corresponding to the input parameter.
	Inputs	orm: ORM_Code
	Outputs	new_srf: SRFT_CELESTIOCENTRIC
	Error conditions	1. INVALID_INPUTCODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.
Operation	Name GetSRFParameters	
	Semantics	This operation outputs the SRF ORM parameter.
	Inputs	source_srf: SRF None
	Outputs	orm ORM_Code
	Error conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. No additional error conditions. (See 11.3.x a.)

Rationale:

1. In accord with comments above.
 2. LifeCycleObject, Inheritance is transitive.
 3. Use type names
- Source: PB " T000"

SEDRIS_T156:

Table 11.12 — 3DLocalSpaceRectangular

Change:

Table 11.12 — 3DLocalSpaceRectangular3D

Class	3DLocalSpaceRectangular3D And throughout.	
Description	An instance of this class corresponds to an instance of SRFT_3D_LOCAL_SPACE_RECTANGULAR	
Superclass	LifeCycleObject, SRF3Dbase	
Operation	Name Create	
	Semantics	Overrides the Create operation on the superclass LifeCycleObject. Creates a 3D local space rectangular SRF corresponding to the input parameters.
	Inputs	up_direction Direction_Of_Up forward_direction Direction_Of_Forward parameters: LSR_3D_Parameters
	Outputs	new_srfSRF3DLocalSpaceRectangular
	Error conditions	1. INVALID_INPUT if input parameters are not valid for this SRF.

Class	3D LocalSpaceRectangular3D And throughout.	
Operation	Name GetSRFParameters	
	Semantics	This operation outputs the SRF parameters.
	Inputs	source_srf ————— SRF None
	Outputs	up_direction ————— Direction_Of_Up forward_direction — Direction_Of_Forward parameters: LSR_3D_Parameters
	Error conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. No additional error conditions (see 11.3.x a.)

Rationale:

1. In accord with comments above.
 2. parameters: LSR_3D_Parameters - Use the defined SRFT parameter types.
 3. Use type names
- Source: PB " T000"

SEDRIS_T157:

Table 11.13 — 2DLocalSpaceRectangular

Change:

Table 11.13 — ~~2D~~LocalSpaceRectangular2D

Class	2D LocalSpaceRectangular2D And throughout.	
Description	An instance of this class corresponds to an instance of SRFT_2D_LOCAL_SPACE_RECTANGULAR	
Superclass	LifeCycleObject, SRF2DBase	
Operation	Name Create	
	Semantics	Overrides the Create operation on the superclass LifeCycleObject. Creates a 2D local space rectangular SRF corresponding to the input parameter.
	Inputs	forward_direction — Direction_Of_Forward parameters: LSR_2D_Parameters
	Outputs	new_srf SRF LocalSpaceRectangular2D
	Error Conditions	1. INVALID_INPUT if input parameter is not valid for this SRF.
Operation	Name GetSRFParameters	
	Semantics	This operation outputs the SRF parameter.
	Inputs	source_srf: ————— SRF None
	Outputs	forward_direction — Direction_Of_Forward parameters: LSR_2D_Parameters
	Error conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. No additional error conditions (see 11.3.x a.)

Rationale:

1. In accord with comments above.
 2. parameters: LSR_2D_Parameters - Use the defined SRFT parameter types.
 3. Use type names
- Source: PB " T000"

SEDRIS_T158:

Table 11.14 — Celestiodetic

Change:

Table 11.14 — Celestiodetic

Class	Celestiodetic	
Description	An instance of this class corresponds to an instance of SRFT_CELESTIODETTIC	
Superclass	LifeCycleObject, SRFwithEllipsoidalHeightBase	
Operation	Name Create	
	Semantics	Overrides the Create operation on the superclass LifeCycleObject. Creates a celestiodetic SRF corresponding to the input parameter.
	Inputs	orm ORM_Code
	Outputs	new_srf SRF Celestiodetic
	Error conditions	1. INVALID_INPUTCODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.
Operation	Name GetSRFParameters	
	Semantics	This operation outputs the SRF ORM parameter.
	Inputs	source_srf: SRF None
	Outputs	orm ORM_Code
	Error conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. No additional error conditions (see 11.3.x a.) [

Rationale:

1. In accord with comments above.
 2. Use type names
- Source: PB " T000"

SEDRIS_T159:

Table 11.15 — 3DLocalTangentPlane

Change:

Table 11.15 — 3DLocalTangentPlaneEuclidean3D

Class	3DLocalTangentPlane Euclidean3D
Description	An instance of this class corresponds to an instance of SRFT_3D_LOCAL_TANGENT_PLANE_EUCLIDEAN3D
Superclass	LifeCycleObject, SRFwithEllipsoidalHeightBase BaseSRFwithTangentPlaneSurface

Class	3DLocalTangentPlaneEuclidean3D	
Operation	Name Create	
	Semantics	Overrides the Create operation on the superclass LifeCycleObject. Creates a 3D local tangent plane SRF corresponding to the input parameters.
	Inputs	orm ORM 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 parameters LTP_Parameters
	Outputs	new_srf SRF3DLocalTangentPlane
	Error Conditions	1. INVALID_INPUT if parameters are not valid for this SRF.
Operation	Name GetSRFParameters	
	Semantics	This operation outputs the SRF ORM parameter.
	Inputs	source_srf: SRF None
	Outputs	orm ORM 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 parameters LTP_Parameters
	Error conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. No additional error conditions. (See 11.3.x a.)

Rationale:

1. Name change: See Clause 8 comment
 2. In accord with comments above.
 3. Use the defined SRFT parameter types.
- Source: PB " T000"

SEDRIS_T160:

Table 11.16 — LocalAzimuthalSphericalTangentPlane

Change:

Table 11.16 — LocalTangentAzimuthalSphericalTangentPlane

Class	LocalTangentAzimuthalSphericalTangentPlane
Description	An instance of this class corresponds to an instance of SRFT_LOCAL_TANGENT_AZIMUTHAL_SPHERICAL_TANGENT_PLANE
Superclass	LifeCycleObject, See [56]. SRFwithEllipsoidalHeightBase BaseSRFwithTangentPlaneSurface

Class	LocalTangentAzimuthalSphericalTangentPlane	
Operation	Name Create	
	Semantics	Overrides the Create operation on the superclass LifeCycleObject. Creates a 3D local azimuthal spherical tangent plane SRF corresponding to the input parameters.
	Inputs	orm ORM geodetic_longitude Long_Float geodetic_latitude Long_Float azimuth Long_Float height_offset Long_Float parameters ATP_Parameters
	Outputs	new_srf SRF LocalTangentAzimuthalSpherical
	Error Conditions	1. INVALID_INPUT if parameters are not valid for this SRF.
Operation	Name GetSRFParameters	
	Semantics	This operation outputs the SRF ORM parameter.
	Input	source_srf: SRF
	Outputs	orm ORM geodetic_longitude Long_Float geodetic_latitude Long_Float azimuth Long_Float height_offset Long_Float parameters LTP_Parameters
	Error conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. No additional error conditions. (See 11.3.x a.)

Rationale:

1. In accord with comments above.

Source: PB " T000"

SEDRIS_T161:

Table 11.17 — SurfaceLocalAzimuthalTangentPlane

Change:Remove.

Rationale: See comments to clause 8.

Source: PB " T000"

SEDRIS_T162:

Table 11.18 — 2DAzimuthal

Change:

Table 11.18 — 2DAzimuthal

Class	2DAzimuthal And throughout.
Description	An instance of this class corresponds to an instance of SRFT_2D_AZIMUTHAL.
Superclass	LifeCycleObject,SRF2Dbase

Class	2DAzimuthal And throughout.	
Operation	Name Create	
	Semantics	Overrides the Create operation on the superclass LifeCycleObject. Creates a 2D azimuthal SRF corresponding to the input parameter.
	Inputs	orm ORM_Code
	Outputs	new_srf SRF_Azimuthal
	Error conditions	1. INVALID_INPUTCODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.
Operation	Name GetSRFParameters	
	Semantics	This operation outputs the SRF ORM parameter.
	Inputs	source_srf: SRF None
	Outputs	orm ORM_Code
	Error conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. No additional error conditions. (See 11.3.x a.)

Rationale:

1. In accord with comments above.

Source: PB " T000"

SEDRIS_T163:

Table 11.19 — LocalCylindricalTangentPlane

Change:

Table 11.19 — LocalTangentCylindricalTangentPlane

Class	LocalTangentCylindricalTangentPlane	
Description	An instance of this class corresponds to an instance of SRFT_SURFACE_LOCAL_CYLINDRICAL_TANGENT_PLANE	
Superclass	LifeCycleObject, SRFwithEllipsoidalHeightBase BaseSRFwithTangentPlaneSurface	
Operation	Name Create	
	Semantics	Overrides the Create operation on the superclass LifeCycleObject. Creates a local cylindrical tangent plane SRF corresponding to the input parameters.
	Inputs	orm ORM geodetic_longitude Long_Float geodetic_latitude Long_Float azimuth Long_Float height_offset Long_Float parameters LTP_Parameters
	Outputs	new_srf SRF_LocalTangentCylindrical
	Error conditions	1. INVALID_INPUT if parameters are not valid for this SRF.

Class	LocalTangentCylindricalTangentPlane	
Operation	Name GetSRFParameters	
	Semantics	This operation outputs the SRF ORM parameter.
	Inputs	source_srf: SRF None
	Outputs	orm ORM geodetic_longitude Long_Float geodetic_latitude Long_Float azimuth Long_Float height_offset Long_Float parameters LTP_Parameters
	Error conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. No additional error conditions. (See 11.3.x a.)

Rationale:

1. In accord with comments above.
Source: PB " T000"

SEDRIS_T164:

Table 11.20 — 2DPolar

Change:

Table 11.20 — 2DPolar

Class	2DPolar And throughout.	
Description	An instance of this class corresponds to an instance of SRFT_2D_POLAR.	
Superclass	LifeCycleObject, SRF2DBase	
Operation	Name Create	
	Semantics	Overrides the Create operation on the superclass LifeCycleObject. Creates a 2D polar SRF corresponding to the input parameter.
	Inputs	orm ORM_Code
	Outputs	new_srf SRF_Polar
	Error conditions	1. INVALID_INPUTCODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.
Operation	Name GetSRFParameters	
	Semantics	This operation outputs the SRF ORM parameter.
	Inputs	source_srf: SRF None
	Outputs	orm ORM_Code
	Error conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. No additional error conditions. (See 11.3.x a.)

Rationale:

1. In accord with comments above.
Source: PB " T000"

SEDRIS_T165:

Table 11.21 — Celestiomagnetic
Table 11.22 — EquatorialInertial
Table 11.23 — SolarEcliptic
Table 11.24 — SolarEquatorial
Table 11.25 — SolarMagnetospheric
Table 11.26 — Solar Magnetic
Table 11.27 — SolarHeliosphericAriesEcliptic
Table 11.28 — SolarHeliosphericEarthEcliptic
Table 11.29 — Solar Heliospheric Earth Equatorial SRF class

Change:

Table 11.21 — Celestiomagnetic

Class	CelestiomagneticSRF	
Description	An instance of this class corresponds to an instance of SRFT_CELESTIOMAGNETIC.	
Superclass	LifeCycleObject, SRFwithEllipsoidalHeightBase Base3DSRF	
Operation	Name Create	
	Semantics	Overrides the Create operation on the superclass LifeCycleObject. Creates a celestiomagnetic SRF corresponding to the input parameter.
	Inputs	orm ORM_Code
	Outputs	new_srf SRF CelestiomagneticSRF
	Error conditions	1. INVALID_INPUTCODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.
Operation	Name GetSRFParameters	
	Semantics	This operation outputs the SRF ORM parameter.
	Inputs	source_srf: SRF None
	Outputs	orm ORM_Code
	Error conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. No additional error conditions. (See 11.3.x a.)

And similarly with:

Table 11.22 — EquatorialInertial
Table 11.23 — SolarEcliptic
Table 11.24 — SolarEquatorial
Table 11.25 — SolarMagnetospheric
Table 11.26 — Solar Magnetic
Table 11.27 — SolarHeliosphericAriesEcliptic
Table 11.28 — SolarHeliosphericEarthEcliptic
Table 11.29 — Solar Heliospheric Earth Equatorial SRF class

Rationale:

1. In accord with comments above.

Source: PB " T000"

SEDRIS_T166:

Table 11.27 — SolarHeliosphericAriesEcliptic

Table 11.28 — SolarHeliosphericEarthEcliptic

Table 11.29 — Solar Heliospheric Earth Equatorial SRF class

Change:Names:

Table 11.27 — ~~Solar~~HeliosphericAriesEcliptic

Table 11.28 — ~~Solar~~HeliosphericEarthEcliptic

Table 11.29 — ~~Solar~~ Heliospheric Earth Equatorial ~~SRF class~~

Rationale: “SolarHelio” is redundant and is not found in clause 8.

Source: PB " T000"

SEDRIS_T167:

Table 11.30 — Mercator

Table 11.31 — ObliqueMercator

Table 11.32 — TransverseMercator

Table 11.33 — LambertConformalConic

Table 11.34 — PolarStereographic

Table 11.35 — Equidistant Cylindrical SRF

Change:

Table 11.30 — Mercator

Class	Mercator	
Description	An instance of this class corresponds to an instance of SRFT_MERCATOR	
Superclass	LifeCycleObject, SRFwithEllipsoidalHeightBase BaseMapProjection	
Operation	Create	
	Semantics	Overrides the Create operation on the superclass LifeCycleObject. Creates a mercator SRF corresponding to the input parameters.
	Inputs	orm — ORM origin_longitude — Long_Float standard_latitude — Long_Float central_scale — Long_Float false_easting — Long_Float false_northing — Long_Float parameters: Mercator_Parameters And similarly throughout.
	Outputs	New_srf SRFMercator And similarly throughout.
	Error Conditions	1. INVALID_INPUT if parameters are not valid for this SRF.

Class	Mercator	
Operation	GetSRFParameters	
	Semantics	This operation outputs the SRF parameters.
	Inputs	source_srf: SRF None
	Output	orm ORM origin_longitude Long_Float standard_latitude Long_Float central_scale Long_Float false_easting Long_Float false_northing Long_Float parameters: Mercator_Parameters
	Error conditions	1. INVALID_SRF if source_srf was not successfully initialised through the API or is invalid. No additional error conditions. (See 11.3.x a.)

And similarly for

Table 0.1 — ObliqueMercator

Table 0.2 — TransverseMercator

Table 0.3 — LambertConformalConic

Table 0.4 — PolarStereographic

Table 0.5 — Equidistant Cylindrical SRF

Rationale:

1. In accord with comments above.

Source: PB " T000"

SEDRIS_T168:

Table 11.36 — BritishNationalGrid

Change:Remove

Rationale: Standard STFs are not classes.They are predefined instances of specific concrete SRFT classes.

Source: PB " T000"

SEDRIS_T169:

11.3.3.x Standard SRFs

Change:Add

11.3.3.x Standard SRFs

The standard SRFs of sub-clause 8.6 are represented as standard instances of the concrete SRFT class corresponding to the SRFT from which each standard SRF is derived.In particular, each standard SRF object is an instance of an SRFT class.A standard SRF object differs from other instances of its SRFT class in that the GetCodes method of the object outputs the corresponding SRF_Code.The function, CreateStandardSRF3D, specified in Table 11.X creates and outputs a standard SRF object from an SRF_Code input.The SRFs objects corresponding to Table 8.30 are specified in Table 11.Y by SRF Label, SRFT class name and SRF creation parameter values.

Table 11.X

Name	CreateStandardSRF3D
Semantic	The input SRF_Code determines a corresponding SRF label and Table 11.Y entries forSRF class and Create parameters.The function creates and outputs a cooresponding SRF object.

Input	code: SRF_Code
Output	new_srf: SRF3D
Error conditions	1. INVALID_CODE, if the code is not a valid SRF_Code. 2. CREATION_FAILURE, if the Create method of the corresponding SRF class fails.

Example: CreateStandardSRF with input code = 3, produces as output an SRF object corresponding to SRF_GEOCENTRIC_EARTH_1984.

Table 11.Y

SRF LABEL	SRF Class	Create parameters
SRF_BRITISH_NATIONAL_GRID	TransverseMercator	Mercator_Parameters = { orm = ORM_OSGB_1936_MEAN_SOLUTION ; origin_longitude = $-2^{\circ}(\pi/180^{\circ})$; standard latitude = $+49^{\circ}(\pi/180^{\circ})$; central_scale = 0,999 601 271 7; false_easting = -400 000 m; false_northing = 100 000 m; }
SRF_DELAWARE_SPCS	TransverseMercator	Mercator_Parameters = { orm = ORM_OSGB_1936_MEAN_SOLUTION ; origin_longitude = $-(75+25/60)^{\circ}(\pi/180^{\circ})$; standard latitude = $+38^{\circ}(\pi/180^{\circ})$; central_scale = $(1 - 1/200\ 000)$; false_easting = 200 000 m; false_northing = 0 m; }
SRF_GEOCENTRIC_EARTH_1984	Celestiocentric	ORM_Code = ORM_WGS_1984;
Etc. to SRF_MARYLAND_SPCS .	Etc.	Etc.

Rationale: Standardized SRF must be treated differently from SRFTs of which they are pre-defined instances.
Source: PB " T000"

SEDRIS_T170:

Table 11.37 — AlabamaSPCS

Table 11.38 — UTM

Table 11.39 — GCS

11.3.3.y SRF Set Classes

Change:

Remove

Table 11.37 — AlabamaSPCS

Table 11.38 — UTM

Table 11.39 — GCS

Add:

11.3.3.y SRF Set Classes

A member of an SRF set for a given ORM is represented as a instance of the SRFT class of the SRF Set. In particular, each SRF Set member object is an instance of an SRFT class. An SRF Set member object differs from other instances of its SRFT class in that the GetCodes method of the object outputs the corresponding SRFS_Code and SRFS_Member_Code. The function CreateSRFSetMember creates and outputs an SRF object in the SRF set corresponding to the input SRFS_Code, ORM_Code and SRFS_Member_Code.

Table 11.Z

Name	CreateSRFSetMember
Semantic	The input SRFS_Code, ORM_Code determine a corresponding SRFS and the SRFS_Member_Code input determines a member of that SRFS. The function creates and outputs an SRF object corresponding to that SRFS member.
Input	set_code: SRFS_Code orm_code: ORM_Code set_member_code: SRFS_Member_Code
Output	new_srf: BaseSRF3D
Error conditions	1. INVALID_CODE, if the set_code is not a valid SRFS_Code, or the orm_code is not a valid ORM_Code for the SRFS, or the set_member_code is not a valid SRFS_Member_Code for the SRFS. 2. CREATION_FAILURE, if the Create method of the corresponding SRF class fails.

Example: CreateStandardSRF with input code = 3, produces as output an SRF object corresponding to SRF_GEOCENTRIC_EARTH_1984.

Rationale: SRF sets are conceptually different from SRFTs.

Source: PB " T000"

Change: Add

11.u Object inheritance hierarchy

The object inheritance hierarchy is summarized in Figure 11.Xa. and b.

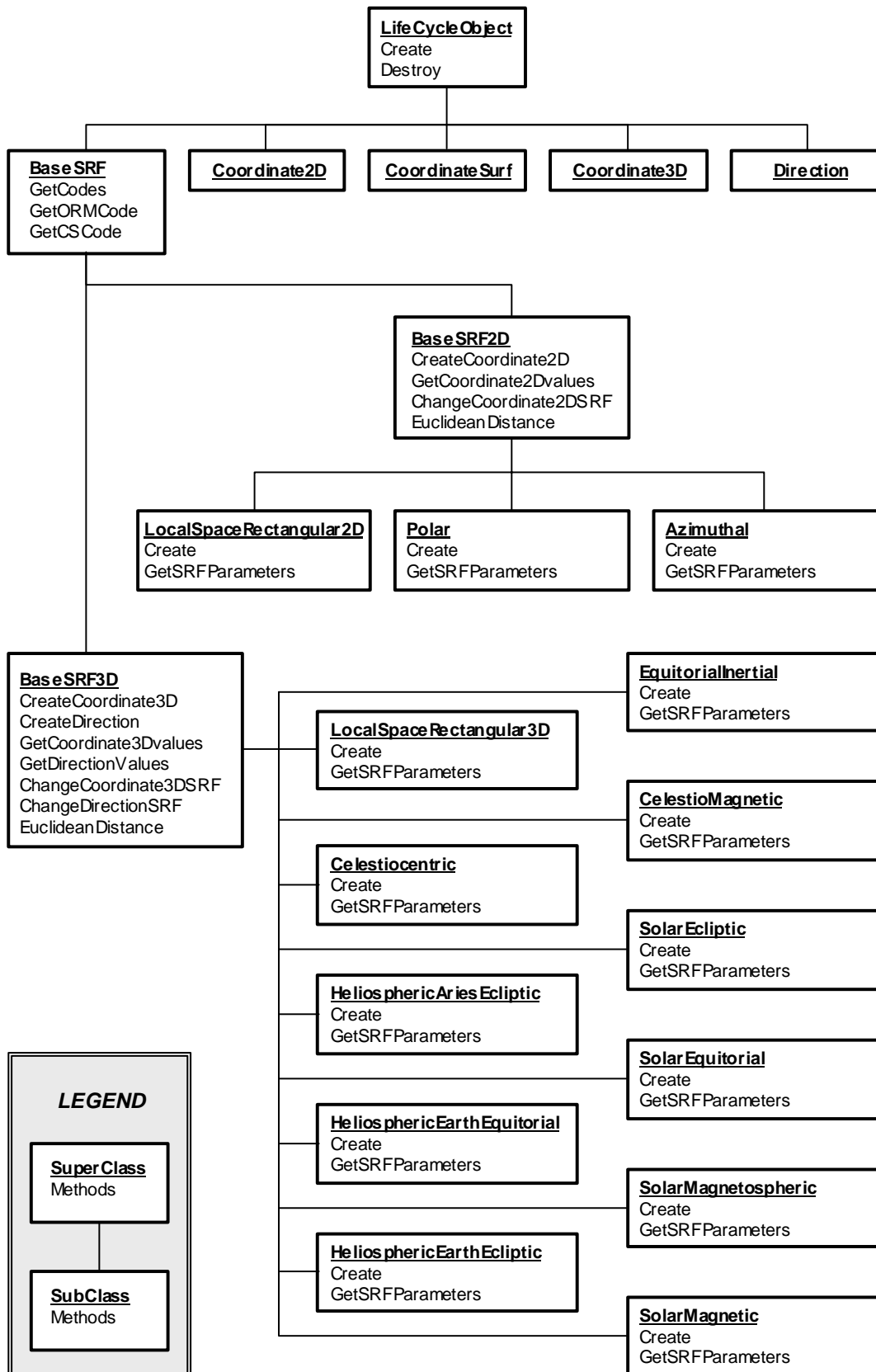


Figure 0.1a — Object inheritance hierarchy

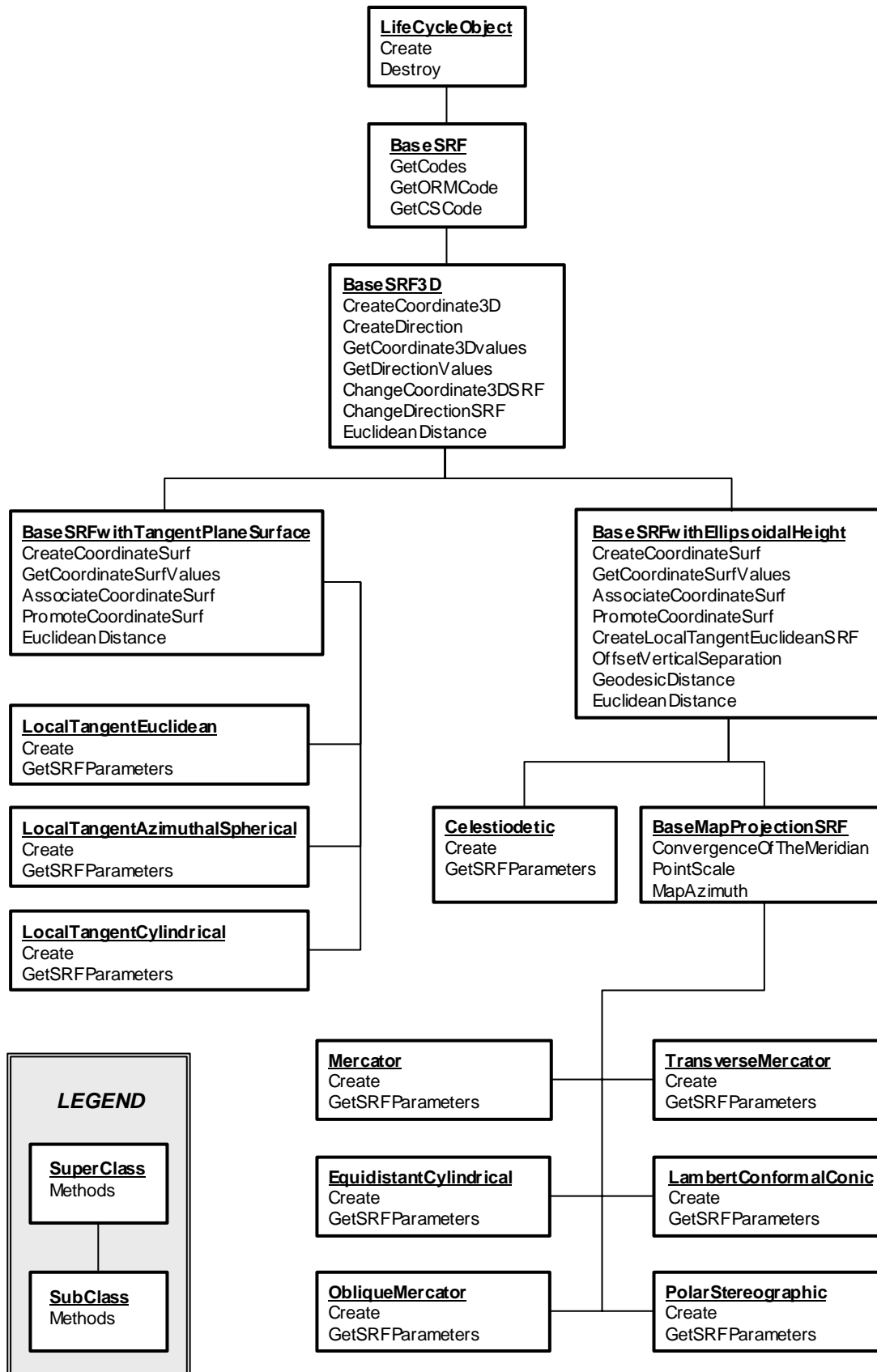


Figure 0.1b — Object inheritance hierarchy

Rationale: Clarity.
Source: PB "T000"

SEDRIS_T172:

11.v Method precedence for Lifecycle objects

Change: Add

11.v Method precedence for Lifecycle objects

There are restrictions on the order in which the methods of Lifecycle objects may be invoked. The restrictions are:

- a. The create method (including subclassed versions) of a Lifecycle object shall be the first method invoked on any instance of the object.
- b. The destroy method (including subclassed versions) of a Lifecycle 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:
 - i. explicit – invoked by the API user,
 - ii. implicit – managed by the runtime system, or
 - iii. 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.

Example 1: Find the Euclidean distance between two locations.

```
--Note: Label in italics denotes a symbolic constant for this example --
Celestiodetic method Create( Input: ORM_N_AM_1983_CONUS; Output: srf)
srf method CreateCoordinate3D( Inputs -77°(π/180°), +38°(π/180°), 0; Output: coordinate1)
srf method CreateCoordinate3D( Inputs +3°(π/180°), +49°(π/180°), 0; Output: coordinate2)
srf method EuclideanDistance( Inputs coordinate1, coordinate2, 0; Output: distance)
-- use distance result --
coordinate1 method destroy
coordinate2 method destroy
srf method destroy
```

Example 2: Change SRF representation of a location from UTM to Celestiocentric

```
--Note: Labels in italics denote symbolic constants for this example --
Function CreateSRFSetMember
( Input: SRFS_UNIVERSAL_TRANSVERSE_MERCATOR, ORM_N_AM_1983_CONUS,
ZONE_23_NORTHERN_HEMISPHERE, Output: source_srf)
Function CreateCoordinate3D( Inputs 120, 400, 0, Output: source_coordinate)
Function CreateStandardSRF( Input: SRF_GEOCENTRIC_EARTH_1984, Output: target_srf)
srf method ChangeCoordinateSRF( Inputs source_srf, source_coordinate, Output: target_coordinate)
-- use result --
source_coordinate method destroy
target_coordinate method destroy
source_srf method destroy
target_srf method destroy
```

Rationale: Need to explain how methods interrelate.

Source: PB " T000"

SEDRIS_T173:

11.4 Non-object functions

Change: Remove

Rationale: See prior comment.
Source: PB " T000"

SEDRIS_T174:

- 11.6.6 SRFs
- 11.6.7 SRF sets
- 11.6.8 SRF set members

Change:Remove

- 11.6.6 SRFs**
- 11.6.7 SRF sets**
- 11.6.8 SRF set members**

Rationale: Redundant with 11.2.5.4 SRF
Source: PB " T000"

Clause 12

SEDRIS_T175:

12.4 Guidelines for labels for registered SRM concepts, d.

Change:

- d. Labels shall contain only uppercase characters (A-Z) with ~~two exceptions~~ one exception:

- ~~1. relational operators ("gt", "lt", "ge", "le", "eq", and "ne"); and~~
2. the radix delimiter symbol "r".

Rationale: Not needed.
Source: PB " T0000"

Clause 13

SEDRIS_T176:

13.1 Introduction, f., g.

Change:

- f. provide conforming operations between SRF realizations of certain pairs of SRF templates; or
- g. provide conforming operations with respect to SRF realizations of a single SRF template.

Rationale: Operations are not performed on templates.
Source: PB " T0000"

SEDRIS_T177:

13.2.1 SRF template conformance, b.

Change:

- b. SRF template F shall be used with the same ~~name~~ label, code, and terminology defined in this International Standard or by registration.

Rationale: "name" is not a specification element for SRFTs (See Table 8.2).
Source: PB " T0000"

SEDRIS_T178:

13.2.1 SRF template conformance, e.

Change:

- e. A shall convert information from an SRF realizing SRF template F to an SRF realizing SRF template G in accordance with ~~Clause 8~~ Clause 10.

Rationale: Information conversion is not specified in clause 8.

Source: PB " T0000"

SEDRIS_T179:

13.3 Conformance of language bindings to the SRM operations API, a.

Change:

- a. L shall implement each function defined in Clause 11 including ~~return~~ output values and error conditions if applicable to Profile P.

Rationale: Clause 11 does not specify return values.L should not need to implement a function that does not apply to any SRFT included in the profile.

Source: PB " T0000"

Annex B

SEDRIS_T180:

Annex B

Change: Add

B.1 Introduction

This informative annex provides some computationally efficient formulae and/or algorithms for some mathematical operations that appear in the body of the text that may be of use in the implementations of the API.

B.2 The Bursa-Wolfe equation

The seven parameter transformation, equation (7.5), involves many multiples of sines and cosines. In many ERM of interest, the rotations are very small.

When a rotation ω is very small, $\sin(\omega) = \omega$ and $\cos(\omega) = 1$ to a very close approximation (when expressed in radians)¹. With this simplification, equation (7.5) applied to transforming embedding E coordinates to reference embedding coordinates can be approximated without any trigonometric functions (in the position vector rotation convention) by the *Bursa-Wolfe equation*:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Reference}} = \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix}_{\text{Reference}} + (1 + \Delta s) \begin{pmatrix} 1 & \omega_3 & -\omega_2 \\ -\omega_3 & 1 & \omega_1 \\ \omega_2 & -\omega_1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_E$$

When the rotations are given in the coordinate frame convention, the rotation terms in the matrix in equation (B.1) reverse their sign.

FOOTNOTE:

1. When $\omega = 1''$, the error multiplied by the radius of the Earth is less than 75 μm . In general, $|\omega - \sin(\omega)| < \frac{1}{2}\omega^2$, when ω is expressed in radians.

Rationale: The Bursa-Wolfe approximation is widely used and is referenced in the text:

7.3.3 Note 2

NOTE 2 The Bursa-Wolfe small rotation approximation of the seven parameter transformation is described in Annex B.

and

10.3.1 Paragraph following Eq. (10.9)

As a consequence, if the difference (Equation (10.9)) between the specified rotation parameters ORM_S and ORM_T are sufficiently small, Equation (7.4) (or equivalently, the Bursa-Wolfe equation (see Annex B) applies.

Source: PB "T0000"

SEDRIS_T181:

Annex B

Change: Add

B.3 – Earth equatorial inertial reference transformation

Given an ERM in the equatorial inertial dynamic binding category, if it is assumed that the ERM z -axis and ORM_WGS_1984 z -axis are coincident, then the transformation, $\mathbf{H}_{\text{EI,WGS84}}(t, \mathbf{p})$, from the ERM embedding to the ORM_WGS_1984 reference ORM embedding is given, at time t , by:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{WGS84}} = \mathbf{H}_{\text{EI,WGS84}} \left(t, \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{EI}} \right) \\ = \begin{pmatrix} \cos(\theta_{\text{GSH}}(t)) & \sin(\theta_{\text{GSH}}(t)) & 0 \\ -\sin(\theta_{\text{GSH}}(t)) & \cos(\theta_{\text{GSH}}(t)) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{EI}}$$

See 7.7.4.1 for notation and terminology.

Rationale: 7.5.2, 4th paragraph states:

Under the assumption that the ORM z -axis and ORM_WGS_1984 z -axis are parallel, the transformation simplifies to a single rotation matrix. This simplified form is described in Annex B.

Source: PB "T0000"

SEDRIS_T182:

Annex B

Change: Add

B.4 – Solar ecliptic reference transformation

In the case of Earth (see[HAPG]), the transformation $\mathbf{H}_{\text{SE,WGS84}}(t, \mathbf{p})$, from solar ecliptic ORM embedding coordinates to ORM_WGS_1984 reference embedding coordinates at time t is:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{WGS84}} = \mathbf{H}_{\text{SE,WGS84}} \begin{pmatrix} t, \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{SE}} \end{pmatrix} \\ = \mathbf{R}_z \mathbf{R}_x \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{SE}}$$

Where:

$$\mathbf{R}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\varepsilon(t)) & -\sin(\varepsilon(t)) \\ 0 & \sin(\varepsilon(t)) & \cos(\varepsilon(t)) \end{pmatrix} \\ \mathbf{R}_z = \begin{pmatrix} \cos(\theta_{\text{GSH}}(t) - \lambda_{\square}(t)) & \sin(\theta_{\text{GSH}}(t) - \lambda_{\square}(t)) & 0 \\ -\sin(\theta_{\text{GSH}}(t) - \lambda_{\square}(t)) & \cos(\theta_{\text{GSH}}(t) - \lambda_{\square}(t)) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rationale: 7.5.3, 2nd paragraph states:

In the case of Earth, the transformation from a solar ecliptic ORM embedding to the ORM_WGS_1984 reference embedding is described in Annex B.

Source: PB "T0000"

Annex E

SEDRIS_T183:

Annex E

Change: WGS72 parameters are wrong. Correct these. See NATO spreadsheet for values of w1=w2=0 w3=0.554.

Rationale: Correction

Source: Meeting "notes"

SEDRIS_T184:

Annex E

Change: Remove duplicate entry code 382. The 2nd entry RIKETS should be 383 (instead of 382), from then on every code should be increased. Also see ORM_RHEA, which has code 381, and should be 382.

Rationale: Correction

Source: Meeting "notes" and CK "T1"

SEDRIS_T185:

Table E.3 — Abstract ORM specifications

Change:

Label	ORM_ABSTRACT_2D	Code	1
Date published	2003	Object name	Abstract 2D

Description	Modelling space	Region	Global
ORM template	ORMT_2D_BI_AXIS_ORIGIN	RD parameterization	
Binding			
Reference type	<i>TBD</i>	References	<i>TBD</i>
Label	ORM_ABSTRACT_3D	Code	42
Date published	2003	Object name	Abstract 3D
Description	Modelling space	Region	Global
ORM template	ORMT_3D_TRI_PLANE	RD parameterization	
Binding			
Reference type	<i>TBD</i>	References	<i>TBD</i>

Rationale: Missing.
Source: PB "T065"

Clause 5 – comments out of sequence.

SEDRIS_T186:

Table 5.12 — Azimuthal spherical

Change:

Generating function formula to:

$$\mathbf{F}(\alpha, \theta, \rho) = (x, y, z),$$

where:

$$x = \rho \cos(\theta) \sin(\alpha),$$

$$y = \rho \cos(\theta) \cos(\alpha), \text{ and}$$

$$z = \rho \sin(\theta).$$

And fix figure

Rationale:

Formula was incorrect.

Parts of the figure are not in the view window.

Source: PB

SEDRIS_T187:

Table 5.28 — Mercator

Table 5.30 — Transverse Mercator

Table 5.31 — Lambert conformal conic

Table 5.32 — Polar stereographic

Table 5.33— Equidistant cylindrical

Mapping equations.

Change:

Simplify to remove Lambda*

$$\Lambda = \lambda - \lambda_{\text{origin}}$$

$$\Lambda^* = \begin{cases} \lambda - \lambda_{\text{origin}} & \text{if } -2\pi \leq \Lambda \leq 2\pi \\ \lambda - \lambda_{\text{origin}} - \pi & \text{if } \Lambda > +2\pi \\ \lambda - \lambda_{\text{origin}} + \pi & \text{if } \Lambda < -2\pi \end{cases}$$

I Rationale:

Of the three cases shown, the domain definition excludes all but one case.

EDITORIAL

Table of Contents

SEDRIS_E001:

Contents, 3 Terms, definitions, symbols, and abbreviated terms

Add a link for Clause 3 (ISO_IEC_18026_E_(E).doc)

Rationale There is no link from the table of contents to Clause 3.
Source NGIT "E001"

Clause 3 – Terms, definitions, symbols, and abbreviated terms

m SEDRIS_E002:

Table 3.3, RDC

Remove this abbreviation from the table.

Rationale This term is no longer used within this standard.
Source NGIT "E002"

SEDRIS_E003:

Table 3.3, RDI

Remove this abbreviation from the table.

Rationale This term is no longer used within this standard.
Source NGIT "E003"

SEDRIS_E004:

3.1 — Notation, last paragraph.

Change:

Move to the beginning of 3.1 –Terms and definitions:

Terms defined in the body of this document are presented in italics at the point where they are defined. The purpose of this notation is to aid readers in determining those sentences that contain the definitions of terms.

Add sentence.

The Index provides a directory of these terms defined in the body of this document.

Rationale: The paragraph is misplaced. The role of the Index is pertinent to this topic.

Source: PB "E001"

SEDRIS_E005:

Table 3.1 — Mathematical notation

Change:

Style	Use	Examples
upper case, bold, italic	vector-valued functions, matrices	<i>F, G, M</i> F, G, M

Rationale: The indicated style is not supported in MathType5

Source: PB "E002"

Clause 4 – Concepts

SEDRIS_E006:

4.2 Spatial objects and object-space, 1st paragraph, 1st sentence

Replace “~~divided into two object types~~” with “divided into two types”.

Rationale: The word “object” is unnecessary here.

Source: NGIT "E007"

SEDRIS_E007:

4.2 Spatial objects and object-space, Example 1

Reword this example. Suggest: “The Sun and the Earth are both celestial objects, each with its own object-space. In the object-space of the Sun, the Sun is fixed and the Earth moves. In the object-space of the Earth, the Earth is fixed and the Sun moves.”

Rationale: The wording of this example is not as clear as it could be.

Source: NGIT "E011"

SEDRIS_E008:

4.2 Spatial objects and object-space, Example 2

Replace “~~spatial object of object type planet~~” with “spatial object of type planet”.

Rationale: The second occurrence of the word “object” is unnecessary.

Source: NGIT "E014"

SEDRIS_E009:

4.2 Spatial objects and object-space, Example 3

Replace “~~components have a location~~” with “each component of the ISS has a location”.

Rationale: The second part of the example is unclear and grammatically incorrect.

Source: NGIT "E015"

SEDRIS_E010:

4.2 Spatial objects and object-space, Example 4, 2nd sentence

Change this sentence to: “The CAD model that specifies the design for that component is defined in an abstract object-space.”

Rationale: This sentence is not clear.
Source: NGIT "E016"

SEDRIS_E011:

4.2 Spatial objects and object-space

Change: Replace 1st paragraph with: “The spatial objects addressed by this International Standard may be divided into two object types: physical objects and abstract objects. Physical objects are objects for which the length of one metre has intrinsic meaning. Abstract objects are objects for which the length of one metre does not have an intrinsic meaning. Virtual, engineering, and/or mathematical models are examples of abstract objects.”

Rationale: The IS is not an animate object (1st sentence). The original 2nd sentence does not add information.
Source: Meeting "E001"

SEDRIS_E012:

4.4 Reference datums, 3rd sentence of paragraph starting with “Figure 4.4 ...”

Reword this sentence. Suggest: “On the right it is bound to a corresponding point in the physical object-space of an object that has been manufactured from that CAD model.”

Rationale: The word order in this sentence makes it appear that it is the object-space, rather than the object, which has been manufactured.
Source: NGIT "E024"

SEDRIS_E013:

Figure 4.4

Replace “(real)” with “(physical)”. Also, consider rearranging the figure so that both object-spaces appear to the right of the position-space.

Rationale: The term “(real)” is not consistent with the terminology used in 4.2. Placing both object-spaces to the right of the position-space would be more consistent with the layout of other figures in this clause.
Source: NGIT "E025"

SEDRIS_E014:

4.5 Object reference models, 1st through 4th paragraphs

Move these paragraphs to the end of 4.4.

Rationale: These paragraphs do not discuss object reference models. They continue to discuss reference datum bindings and normal embeddings of position-space into object-space.
Source: NGIT "E026"

SEDRIS_E015:

4.5 Object reference models, 1st paragraph, 1st sentence

Reword this sentence. Suggest that “reference datum binding” be defined first, then a “bound reference datum” can be defined simply as a reference datum that has such a binding.

Rationale: It is not clear what the phrase “reference datum binding” refers to. In 4.4, the phrase “bound reference datum” was defined, but it is not clear that these two phrases are equivalent. What does a “reference datum binding” consist of? Also, the phrase “geometric construction of the binding” is not clear. Previously, the phrase “geometric construct” was used to refer to the representation of a reference datum in both position-space and object-space.

Source: NGIT "E027"

SEDRIS_E016:

4.5 Object reference models, 1st paragraph, 1st sentence

State that, in general, a reference datum binding (or should that be a bound reference datum?) is compatible with many different normal embeddings. By carefully combining multiple reference datum bindings, a single compatible normal embedding can be specified.

Rationale: The key point is not that a single compatible normal embedding may or may not exist, but that in general, many compatible normal embeddings exist, and the goal is to combine multiple reference datums in such a way as to identify a single, unique compatible embedding.

Source: NGIT "E028"

SEDRIS_E017:

4.5 Object reference models, Example 1, 1st sentence

Change “~~reference datum origin point~~” to “origin point reference datum”.

Rationale: The phrase “reference datum origin point” is not consistent with the phrase “*x*-axis reference datum” in the next sentence, and other similar terms in this clause. In general, it appears better to state reference datum types as prefixes rather than suffixes.

Source: NGIT "E029"

SEDRIS_E018:

4.5 Object reference models, Example 1, 4th sentence

Change “~~lies in the directed line~~” to “lies on the directed line”.

Rationale: The phrase “lies in the directed line” is not consistent with the phrasing in the rest of this paragraph.

Source: NGIT "E030"

SEDRIS_E019:

4.5 Object reference models, Example 1, last sentence

Change “~~there will be~~” to “there are”.

Rationale: The tense of the phrase “there will be” is not consistent with the tense of the previous sentences.

Source: NGIT "E031"

SEDRIS_E020:

4.5 Object reference models, 11th paragraph (immediately preceding Example 4)

Reword the beginning of this sentence. Suggest: “An object reference model is realized from an object reference model template by ...”.

Rationale: The beginning of this sentence is awkward.
Source: NGIT "E032"

SEDRIS_E021:

4.5 Object reference models, Example 4

Explain why what Example 2 identified as an object reference model, is now being called an object reference model template. Identify the template specifically as “ORMT_3D_OBLATE_-SPHEROID”.

Rationale: The thing that was described in Example 2 as an “object reference model” is here identified as being an “object reference model template”. It should not be identified first as one type of thing, and later as a different type of thing, at least not without further explanation.
Source: NGIT "E033"

SEDRIS_E022:

4.5 Object reference models, Example 5

Replace “~~World Geodetic System 1984~~” with “World Geodetic System 1984 (specified in Annex E as ORM WGS 1984)” and move the link from the name to the identifier.

Rationale: This example is not consistent with the following example (Example 6) in how it uses identifiers for object reference models.
Source: NGIT "E034"

SEDRIS_E023:

4.6.1 Abstract coordinate systems, 3rd paragraph, 1st sentence

Replace “~~coordinate system function~~” with “coordinate system generating function”.

Rationale: The phrase “coordinate system function” is not consistent with the introduced earlier in this subclause.
Source: NGIT "E035"

SEDRIS_E024:

4.6.1 Abstract coordinate systems, 6th paragraph, 1st sentence

Replace “~~coordinate system function~~” with “coordinate system generating function”.

Rationale: The phrase “coordinate system function” is not consistent with the terminology defined earlier in this clause.
Source: NGIT "E036"

SEDRIS_E025:

Figure 4.7, caption

Replace “~~coordinate system function~~” with “coordinate system generating function”.

Rationale: The phrase “coordinate system function” is not consistent with the terminology defined earlier in this clause.

Source: NGIT "E037"

SEDRIS_E026:

Figure 4.9

Add the geodetic latitude and longitude angles, and the geodetic height, to the figure.

Rationale: This figure does not show the angles that define latitude and longitude, nor the distance above/below the surface of the oblate spheroid that defines geodetic height.

Source: NGIT "E038"

SEDRIS_E027:

4.6.3 Spatial coordinate systems, 1st paragraph, 2nd sentence

Reorder this sentence. Suggest replacing “~~a position-space normal embedding E and an abstract coordinate system for that position-space~~” with “an abstract coordinate system that maps n -tuples in a coordinate-space to positions in position-space, and a normal embedding that maps those positions in position-space to points in an object-space”.

Rationale: The ordering of the components of a spatial coordinate system in this sentence does not describe a logical sequence from coordinate-space to position-space to object-space.

Source: NGIT "E039"

SEDRIS_E028:

Figure 4.10

Replace the label “~~3D orthonormal embedding~~” with “normal embedding”.

Rationale: The wording of the label “3D orthonormal embedding” is not consistent with the wording used in the preceding paragraph. Adding “3D” and “ortho-“ disrupts the correspondence between the text and the figure.

Source: NGIT "E040"

SEDRIS_E029:

4.6.3 Spatial coordinate systems, Example 1

Replace “~~distance~~” with “distinct”.

Rationale: The phrase “two distance spatial Euclidean 3D coordinate systems” is incorrect.

Source: NGIT "E041"

SEDRIS_E030:

4.6.3 Spatial coordinate systems, Example 2

Remove this example.

Rationale: This example is not useful. The coordinate-space, position-space, and object-space are all completely generic. The fact that all three of these spaces are described as being \mathbf{R}^3 is confusing. It is not clear what it means for 3-tuples in these three spaces to be “the same”.

Source: NGIT "E042"

SEDRIS_E031:

4.7 Spatial reference frames, c (1st occurrence)

Replace this list item with: “a binding of the abstract coordinate system parameters, if any, to specific values”.

Rationale: The omission of “abstract” before coordinate system, when it is included in all other nearby instances, is confusing. Also, it is not clear what the abstract coordinate system parameters are bound to.
Source: NGIT "E043"

SEDRIS_E032:

4.7 Spatial reference frames, 1st paragraph

Clarify how the specific spatial object that the SRF is based on is specified. Suggest adding it as an initial entry in the list that follows this paragraph.

Rationale: It is not clear where the specific spatial object (e.g. Earth) is identified. Should it be part of the SRF? Or is it implied by the SRF's ORM?
Source: NGIT "E044"

SEDRIS_E033:

4.7 Spatial reference frames, 3rd paragraph

Replace “~~set of specifications of spatial reference frames~~” with “set of spatial reference frames”. Replace “~~that share the same abstract coordinate system, the object reference model components of which are realizations of the same object reference model template, and that model spatial objects of the same spatial object type~~” with “that share the same underlying spatial object type, the same underlying object reference model template, and the same abstract coordinate system type”

Rationale: The wording of this long sentence is awkward, and the order in which the criteria are listed does not match the following list.
Source: NGIT "E045"

SEDRIS_E034:

4.7 Spatial reference frames, a (2nd occurrence)

Change “~~an object type~~” to “a spatial object type”.

Rationale: As stated, this list item is too general.
Source: NGIT "E046"

SEDRIS_E035:

4.7 Spatial reference frames, b (2nd occurrence)

Replace “~~constraint~~” with “template”.

Rationale: The word “constraint” appears to be incorrect here.
Source: NGIT "E047"

SEDRIS_E036:

4.7 Spatial reference frames, c (2nd occurrence)

Delete “specification”.

Rationale: The word “specification” here is extraneous and confusing, as it suggests that an “abstract coordinate system specification” is a distinct concept from an “abstract coordinate system”. In general, avoid using previously defined noun phrases as adjectives.

Source: NGIT "E048"

SEDRIS_E037:

4.7 Spatial reference frames, 5th paragraph (immediately preceding Example 1)

Reword this sentence. Suggest: “A spatial reference frame is realized from a spatial reference frame template by identifying a spatial object of the type specified by the template, identifying an object reference model that is a realization of the object reference model template specified by the template, and the abstract coordinate system parameters, if any, are bound to specific values.”

Rationale: The intent of this sentence is not clear. It reads as though the goal is to identify the template that a particular SRF was realized from. Instead, it should describe how to realize an SRF from an SRF template.

Source: NGIT "E049"

SEDRIS_E038:

4.7 Spatial reference frames, 9th paragraph (immediately following Example 3), and a (in following list)

Remove these phrases, and instead add an item to the list: “use the same object reference model”. It may also be necessary or useful to specify that the SRFs share the same spatial object.

Rationale: The phrase “Given an object reference model,” obscures the purpose of this sentence, which is to define the concept of “spatial reference frame set”. In list item a, the phrase “using the given object reference model” should be stated as a separate list item. (Again, it is not completely clear here whether an ORM specifies the spatial object, or the SRF does.)

Source: NGIT "E050"

SEDRIS_E039:

4.7 Spatial reference frames, 9th paragraph (immediately following Example 3), b (in following list)

Replace “the valid regions of the set members are non-overlapping” with “have non-overlapping valid regions”. Consider adding a brief definition of the valid region of an SRF.

Rationale: The wording of this list item is awkward. Also, the concept of the valid region of an SRF has not previously been discussed.

Source: NGIT "E051"

SEDRIS_E040:

4.7 Spatial reference frames, 10th paragraph (immediately preceding Example 4)

Replace “~~spatial reference frame set specification~~” with “spatial reference frame set”.

Rationale: The word “specification” here is extraneous and confusing, as it suggests that a “spatial reference frame set specification” is a distinct concept from a “spatial reference frame set”. Avoid using previously defined noun phrases as adjectives.

Source: NGIT "E052"

SEDRIS_E041:

4.9 Spatial operations, 1st paragraph, 2nd sentence

Replace “~~coordinates~~” with “positions”.

Rationale: The word “coordinates” here is not consistent with the phrasing used in the Introduction and Scope clauses.

Source: NGIT "E053"

SEDRIS_E042:

4.9 Spatial operations, 2nd paragraph, 6th sentence

Replace “~~coordinate system function~~” with “coordinate system generating function”.

Rationale: The phrase “coordinate system function” is not consistent with the terminology defined earlier in this clause.

Source: NGIT "E054"

SEDRIS_E043:

4.11 Registration and 4.13 Concept specification

Combine 4.11 and 4.13.A suggested outline is:

4.11 Registration

4.11.1 Registration procedure (currently 4.11)

4.11.2 Standardized and registered concepts (currently 4.13.1)

4.11.3 Specification fields (currently 4.13.2)

Rationale: The content of 4.13 is directly related to the content of 4.11. Both deal with the extension of concept sets through registration. Both refer to clause 12. The presence of 4.12 between them disrupts the logical flow of this clause.

Source: NGIT "E055"

Clause 5

SEDRIS_E044:

5.2.2 Abstract CS; 2nd and 3rd sentences of the 2nd bullet C

Change: Both sentences are missing an “of”, as in “...shall be a subset of an implicitly ...”

Rationale: Correctness.

Source: Meeting “notes”

SEDRIS_E045:

Clause 5, Section 5.3.5.1: Inconsistent symbol used for easting in second sentence. Should be consistent with Figure 5.4

Change: “...easting coordinate *n* is the...” to “easting coordinate *u* is the...”

Rationale: Correctness

Source: NIMA "E009a"

SEDRIS_E046:

Clause 5, Table 5.30: Proper reference to elliptic functions

Change: “...Jacobian elliptic...” to “...Jacobi elliptic...”

Rationale: Correctness

Source: NIMA "E009b"

SEDRIS_E047:

Bibliography: Missing entries for LLEE and DOZI referenced in Table 5.30.

Change: Add references for LLEE and DOZI.

Rationale: Completeness

Source: NIMA "E021"

Clause 7

SEDRIS_E048:

7.3.3 - Specification of 3D normal embeddings, 3rd sentence

Change:

In general, the origin of the E2 embedding may be displaced with respect to the origin of the E1 embedding and the axes of the E2 embedding may also be rotated and/or differently scaled with respect to the axes of the E1 embedding.

Rationale: Typo

Source: PB "E0000"

Clause 8

SEDRIS_E049:

8.1 Introduction, paragraph 1. Sentence 1.

Change: A spatial reference frame is a spatial coordinate system for a region of ~~the space of an~~ object-space formed by the binding of an abstract coordinate system to the linear embedding specified by an ORM for that object.

Rationale: Consistency and clarity.

Source: PB "E003"

SEDRIS_E050:

8.1 Introduction, paragraph 1. last sentence, and throughout.

Change: In particular, a CS based on an oblate spheroid (or sphere) must match the parameters of the oblate spheroid ~~(or sphere)~~ RD of the ORM.

Rationale: "(or sphere)" is included in oblate spheroid RD definition.

Source: PB "E004"

SEDRIS_E051:

8.5.7 through 8.5.20

Change:

The order of the SRF sub clauses does not match the order in Table 8.3 — SRFT directory. Change the order to match grouping the SRFT specifications under level 3 sub clauses:

8.5.2 3D SRFTs

8.5.3 Map projection based SRFTs

8.5.4 2D SRFTs, or

explain ordering rule.

Rationale: The current ordering rule is not obvious.

Source: PB "E005"

SEDRIS_E052:

8.6 SRFs, 2nd paragraph

Change:

The specification elements for ~~SRFTs~~ SRFs are defined in Table 8.29.

Rationale: Typo.

Source: PB "E006"

Clause 9

SEDRIS_E053:

Table 9.2, VOS_IGLD_1955 and VOS_NAVD_1988, Notes

VOS_IGLD_1955, Notes

“at Pointe-au-Pere, Quebec, on the Gulf of St. Lawrence”

VOS_NAVD_1988, Notes

“~~at Father Point, Rimouski, Quebec, Canada~~ at Pointe-au-Pere, Quebec, on the Gulf of St. Lawrence”

Rationale: Both of these refer to the exact same place. Father Point is English for Pointe-au-Pere. It should be written the same way for both notes to avoid confusion. The first (Pointe-au-Pere) is preferable/proper. Rimouski is only mentioned to help people with the locale since Pointe-au-Pere is a small village. Canada is not needed since it is obvious it is Canada.

Source: CH "E001"

Clause 10

SEDRIS_E054:

10.3.1 - Object fixed ORM realizations for a single object, 1st paragraph last sentence.

Change:

This form of vector operation is an ~~affine~~ affine operation

Rationale: The term “affine” is defined in Annex A.
Source: PB "E0000"

SEDRIS_E055:

10.5.2 - Canonical local tangent plane (CLTP), 1st paragraph last sentence.

Change:

The CLTP Y-axis points in the direction of the tangent vector of the latitude coordinate curve at the LTP origin ($\lambda, \phi, 0$) and the CLTP Z-axis points in the ~~directions~~ direction of the ORM oblate spheroid RD surface normal at that point.

Rationale: Grammar.
Source: PB "E0000"

Annex C – UML Diagrams for SRM Concepts

SEDRIS_E056:

Figure C.1

Add an additional level of abstract subclasses under `OrientedSurface` that includes `Plane`, `OblateSpheroid`, `Sphere`, and `TriAxialSpheroid`. List the individual concrete classes under these abstract classes, as appropriate.

Rationale: These “sub-categories” under `OrientedSurface` should not be shown as labels on each branch of the class hierarchy, but rather as an additional level of abstract classes. This would allow the ORM templates shown in Figure C.2 to reference these abstract classes.

Source: NGIT "E056"

SEDRIS_E057:

Figure C.2

Replace the names “sphere RD”, “oblate spheroid RD”, and “prolate spheroid RD” with the abstract class names “`Sphere`”, “`OblateSpheroid`”, and “`ProlateSpheroid`”, respectively. Shade these components to indicate that they are abstract classes.

Rationale: These names are not consistent with classes, abstract or concrete, shown in Figure C.1. Creating class names for these abstract classes will allow the ORM templates shown in Figure C.2 to properly reference the appropriate abstract classes in Figure C.1.

Source: NGIT "E057"

SEDRIS_E058:

Figures C.1 and C.2

Add the abstract class “`ProlateSpheroid`” to Figure C.1, along with a list of concrete sub-classes from Table D.4 (if any). Add the `ORMT_3D_TRI_AXIAL_SPHEROID` template to Figure C.2.

Rationale: There is no ORM template shown in Figure C.2 that uses a tri-axial spheroid `OrientedSurface` RD. Conversely, there are no `OrientedSurface` RDs shown in Figure C.1 that match the “prolate spheroid RD” component class shown in Figure C.2.

Source: NGIT "E058"



SEDRIS_E059:

Figures C.4

Complete this figure, replacing all instances of “Etc.” with the other abstract coordinate systems defined in Clause 5.

Rationale: This figure does not fully illustrate the hierarchical classes of abstract coordinate systems supported by the SRM.

Source: NGIT "E059"

US_G001:
US_G006
TECHNICAL
Clause 2: Normative References
Clause 3 Terms, definitions, symbols, and abbreviated terms
US_T002:
Clause 4 Concepts
Clause 5 Coordinate systems
5.1.1 Introduction
Clause 6 Temporal Coordinate Systems
Clause 9
Clause 10
Clause 12
Annex E
Annex J
EDITORIAL 10	
Scope 10	
Clause 2 Normative references
Clause 3 Terms, definitions, symbols, and abbreviated terms
Clause 4 Concepts
US_E011:
US_E014:
Clause 5 Coordinate systems
US_E019:
US_E021:
US_E023:
5.2.2, notes and footnotes
5.2.2, note 3
5.2.4.2, Example 1
US_E026:
US_E030:
US_E032:
US_E034:
US_E035:
Clause 6 Temporal coordinate systems
6.2.1, Examples
US_E042:
Clause 7
Table 7.1, Directed Curve
US_E045:
US_E047:
Equation 7.2
US_E048:
US_E051:
US_E055:
US_E056:
Note 2 after Table 7.10
US_E059:

7.5.5, 17

Clause 8	17
Clause 9	
Clause 10	17
Sentence after equation 10.2	
US_E065:.....	
Sentence after equation 10.8	
US_E069:.....	
Sentence after equation 10.9	
US_E070:.....	
10.3.3, last sentence	
US_E071:.....	
Note before equation 10.15	
US_E072:.....	
Sentence after equation 10.19	
US_E073:.....	
Paragraph before 10.4.2	
Sentence after equation 10.31	
Sentence after equation 10.32	
Sentences before and after equation 10.33.....	
Sentence after Figure 10.3	
Sentence after Figure 10.3	
10.5.1, last paragraph	
US_E083:.....	
US_E085:.....	
Clause 11	
11.1, last sentence	
US_E092:.....	
11.2.6.3.1 and 11.2.6.3.2	
Table 11.7, SRF3Dbase, semantics	
Table 11.7, GetCoordinate3DValues, error conditions 2.....	
Table 11.7, ChangeDirectionSRF, semantics.....	
Table 11.7, last row, last item	
Table 11.8, Create2DCoordinate, semantics	
US_E101:.....	
Table 11.8, Create2DCoordinate, semantics.....	
Table 11.8, last Abstract operation, Semantics	
US_E104:.....	
Table 11.22, semantics.....	
Table 11.33, semantics.....	
Table 11.35, semantics.....	
Table 11.37, semantics.....	
Clause 12	
US_E115:.....	
US_E116:.....	
12.4 p 23	
Clause 13	24
Annex A	24
A.4 c	24
A.6.2, paragraph before note	
Annex C	24
Annex E	25
Table E.4, Label ORM_ADINDAN_ETHIOPIA, last column	
Table E.4	
Table E.4	
Table E.4	
Annex F	26
Table F.4	
Table F.4	

Annex G	26
G.1	26
G.3.1 b	27
Annex H	27
H.4	27
H.9, last sentence	2
Annex J	27
US_E142:.....	2
Table J.7, ORM_N_AM_1927_EASTERN_US, Description.....	2
Alphabetical Index.....	2
Lambert, 2 entries.....	2
Bibliography	27
83582	27
General.....	2
Entire document	2
UK_G004:	2
Entire document	2
UK_G007:	2
Technical	2
Table of Contents	2
Entry for Clause 3.....	2
Table of tables. This table of tables is incomplete. There are many missing tables including all in Annex E. If there is to be a table of tables, it should be complete.....	2
Foreword.....	2
Introduction	2
Clause 1—Scope	2
Clause 2—Normative references	2
Clause 3—Terms, definitions, symbols and abbreviated terms	2
Clause 4—Concepts	2
Clause 5—Coordinate systems.....	2
Clause 6— Temporal coordinate systems	2
Clause 7—Reference datums, embeddings and object reference models.....	2
Clause 8—Spatial reference frames	2
Clause 9—Vertical offset surfaces.....	2
Clause 10—Spatial operations.....	2
Clause 11— Application program interface	2
Clause 12— Registration.....	4
Clause 13— Conformance	4
Annex A—Mathematical foundations	4
A.1, first sentence.....	4
Annex B— Implementation notes.....	4
Annex C— UML diagrams for SRM concepts	4
Annex D— RDs associated with celestial objects	4
Appendix E—ORMs.....	4
Appendix F— Abbreviations and acronyms used in the construction of labels	4
F	48
Appendix G— Change and deprecation plan	4
Annex H —Templates for registration proposals	4
Annex I —Conformance testing for spatial operations	4
Appendix J— Deprecated SRM concepts	4
Bibliography	4
Editorial.....	5
Introduction	5
Clause 3— Normative references	5
Clause 3—Terms, definitions, symbols and abbreviated terms	5
Clause 4—Concepts	5
Clause 5—Coordinate systems.....	5
Clause 6—Temporal coordinate systems	5

Clause 7—Reference datums and object reference models	5
Clause 9— Vertical offset models.....	5
Clause 10— Spatial operations.....	5
Clause 11—Application program interface	5
Annex A—Mathematical foundations	5
Annex C UML diagrams for SRM concepts.....	5
Annex D— RDs associated with celestial objects	5
Annex E—ORMs	5
GENERAL.....	59
Clause 4	59
TECHNICAL	60
Clause 3 – Terms, definitions, symbols, and abbreviated terms	60
a SEDRIS_T001:.....	6
Clause 4 – Concepts.....	61
Clause 5 Coordinate systems	67
b and.....	6
Clause 7	72
Table 7.10 — ORM specification fields	7
Clause 8	75
Table 8.3 — SRFT directory	7
5.5.x Planetodetic SRF	7
Table 8.6 — Local space rectangular 2D SRF template	8
Table 8.8 — Local tangent plane 3D SRFT	8
Table 8.9 — Local azimuthal spherical tangent plane SRFT	8
SEDRIS_T076:	82
Table 8.11 — Azimuthal 2D SRFT	8
c Specification element	8
d Value.....	8
e Description	8
f Azimuthal 2D SRF. An SRF for 2D abstract space based on the azimuthal CS.	8
Table 8.12 — Local cylindrical tangent plane SRFT.....	8
Table 8.13 — Polar 2D SRFT.....	8
Clause 9	87
g 9.2 - Object reference surfaces	8
h 9.2.2 - Reference spheroids (and spheres)	8
i 9.2.6 – Vertical offset height and elevation	8
9.2.7 Other vertical coordinate definitions.....	9
j 9.4 Other vertical coordinate definitions	9
Clause 10	90
Clause 11	90
Direction Of Forward	9
Direction Of Up	9
11.2.5.4 SRFT, SRF, and SRFS and SRFS member codes	9
11.2.6.2.5 OM Oblique Mercator Parameters	9
Table 11.9 — BaseSRFwithEllipsoidalHeightBase	114
k 11.3.3.w BaseMapProjection Class	11
Table 11.11 — Celestiocentric	121
Clause 12	137
Clause 13	137
Annex B.....	138

B.1 Introduction.....	13
B.2 The Bursa-Wolfe equation.....	13
B.3 – Earth equatorial inertial reference transformation.....	13
B.4 – Solar ecliptic reference transformation	13
Annex E.....	140
I Rationale:	14
EDITORIAL.....	142
Table of Contents.....	142
Clause 3 – Terms, definitions, symbols, and abbreviated terms.....	142
m SEDRIS_E002:.....	14
Clause 4 – Concepts.....	143
Clause 5	150
Clause 7	151
Clause 8	151
Clause 9	152
Clause 10	152
Annex C – UML Diagrams for SRM Concepts	153
Application program interface	16
Introduction	16
Non-object data types	16
Overview	16
Numbers.....	16
Object_Reference	16
Enumerated data types	16
Introduction	16
Axis_Direction	16
Polar_Aspect.....	16
Coordinate_Valid_Region	16
Selection data types.....	16
Introduction	16
CS_Code 164	
ORM_Code 164	
VOS_Code 164	
SRFT_Code.....	16
SRF_Code 165	
SRFS_Code and SRFS_Member_Code	16
Status_Code	16
Structured data types.....	16
Introduction	16
SRFT parameters.....	16
LSR_3D_Parameters 166	
LSR_2D_Parameters 166	
LocalTangentEuclidian_Parameters	16
LocalTangent_Parameters 167	
Mercator_Parameters 167	
Oblique_Mercator_Parameters	16
LCC_Parameters 167	
PS_Parameters 167	
EC_Parameters 168	
Object types.....	16
Introduction	16

Class Specification Format.....	16
LifeCycleObject	16
Private Objects	17
Introduction	17
Coordinate3D.....	17
Coordinate2D.....	17
CoordinateSurf	17
Direction 170	
Abstract classes	17
BaseSRF 171	
BaseSRF2D171	
BaseSRF3D173	
BaseSRFwithTangentPlaneSurface.....	17
BaseSRFwithEllipsoidalHeight.....	17
BaseMapProjection	18
SRF concrete subclasses of BaseSRF2D	18
LocalSpaceRectangular2D.....	18
Azimuthal 183	
SRF concrete subclasses of BaseSRF3D	18
LocalSpaceRectangular3D.....	18
Celestiocentric.....	18
CelestioMagnetic	18
EquatorialInertial	18
SolarEcliptic	18
SolarEquatorial.....	18
SolarMagnetospheric	18
SolarMagnetic.....	18
HeliosphericAriesEcliptic.....	18
HeliosphericEarthEcliptic	18
HeliosphericEarthEquatorial.....	19
SRF concrete subclasses of BaseSRFwithTangentPlaneSurface.....	19
LocalTangentEuclidean	19
LocalTangentAzimuthalSpherical	19
LocalTangentCylindrical.....	19
SRF concrete subclasses of BaseSRFwithEllipsoidalHeight	19
Celestiodetic.....	19
SRF concrete subclasses of BaseMapProjection.....	19
ObliqueMercator	19
TransverseMercator	19
LambertConformalConic.....	19
PolarStereographic	19
Equidistant Cylindrical.....	19
Standard SRFs.....	19
SRF Set Classes	19
Object inheritance hierarchy	19
Method precedence for life cycle objects	20

Table 11.1 — Minimum value ranges for integer data types	16
Table 11.2 — LifeCycleObject.....	16
Table 11.3 —Coordinate3D	17
Table 11.4 — Coordinate2D	17
Table 11.5 —CoordinateSurf	17
Table 11.6 — Direction.....	17

Table 11.7 — BaseSRF.....	17
Table 11.8 — BaseSRF2D	17
Table 11.9 — BaseSRF3D	17
Table 11.10 — BaseSRFwithEllipsoidalHeight.....	17
Table 11.11 — BaseMapProjection.....	18
Table 11.12 — LocalSpaceRectangular2D	18
Table 11.13 — Azimuthal	18
Table 11.14 — Polar.....	18
Table 11.15 — LocalSpaceRectangular3D	18
Table 11.16 — Celestiocentric.....	18
Table 11.17 — CelestioMagnetic	18
Table 11.18 — EquatorialInertial	18
Table 11.19 — SolarEcliptic	18
Table 11.20 — SolarEquatorial.....	18
Table 11.21 — SolarMagnetospheric.....	18
Table 11.22 — SolarMagnetic	18
Table 11.23 — HeliosphericAriesEcliptic	18
Table 11.24 — HeliosphericEarthEcliptic.....	18
Table 11.25 — HeliosphericEarthEquatorial	19
Table 11.26 — LocalTangentEuclidean	19
Table 11.27 — LocalTangentAzimuthalSpherical.....	19
Table 11.28 — LocalTangentCylindrical	19
Table 11.29 — Celestiodetic.....	19
Table 11.30 — Mercator	19
Table 11.31 — ObliqueMercator.....	19
Table 11.32 — TransverseMercator.....	19
Table 11.33 — LambertConformalConic.....	19
Table 11.34 — PolarStereographic.....	19
Table 11.35 — Equidistant Cylindrical	19
Table 11.36 — CreateStandardSRF3D.....	19

Application program interface

Introduction

This International Standard specifies an API for the spatial operations in [Clause 10](#). The API specifies non-object data types (see [0](#)), and classes (object types) (see [0](#)) used to perform the spatial operations.

Class is the term used to express the type of an *object*. Each class definition specifies the methods (if any) on the object. *Methods* are specified by giving their syntax (input and output parameters), semantics (how the inputs interact with the state of the object and produce any outputs), and error conditions. In particular, the state of an object 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 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 called the *object life cycle*. Objects inherit methods from other objects through the *subclass/superclass* relationship. Method inheritance is transitive: A subclass also inherits the method that have been inherited by its superclass.

The API includes both object data types (classes) and non-object data types. 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 five classes that expose no operations: one SRF class; three coordinate classes: [Coordinate2D](#), [Coordinate3D](#), and [CoordinateSurf](#); and one direction class: [Direction](#) (see [0](#)). These classes are private types which hide all aspects of the implementation of instances of these objects from the application.

The API specifies seven abstract classes: [LifeCycleObject](#), [BaseSRF](#), [BaseSRF2D](#), [BaseSRF3D](#), [BaseSRFwithEllipsoidalHeight](#), [BaseSRFwithTangentPlaneSurface](#), and [BaseMapProjection](#) (see [0](#)). These abstract classes are used as the base classes from which subclasses including concrete classes inherit a common set of methods. [LifeCycleObject](#) includes the creation and destruction methods which all other classes inherit.

The API specifies a set concrete classes that implement specific SRFs specified in [Clause 8](#) (see [0](#)). The class hierarchy is illustrated in [Figure 0.1](#).

Non-object data types

Overview

Basic non-object data types represent single pieces of information such as numbers, alphanumeric characters, Booleans, and other individual data items.

Numbers

Two categories of numbers are specified: integer numbers and floating point numbers.

Two data types for integer numbers are specified. One is a general-purpose integer data type and the other a non-negative integer data type. These are intended to correspond to signed and unsigned integer data types in programming languages. The general-purpose integer data types are `Byte` and `Short_Integer` and the non-negative integer data type is `Short_Unsigned_Integer`.

All implementations which conform to this standard shall support at least the minimum ranges for values of these data types as specified in [Table 0.1](#).

Table 0.1 — Minimum value ranges for integer data types

Data type	Value range
Byte	[-128, 127]
Short_Integer	[-32767, 32767]
Short_Unsigned_Integer	[0, 65535]

Long_Float is a non-object data type defined for floating point. This data type corresponds to the double precision floating point type defined by the IEEE standard for floating point numbers [754]. However, implementations on architectures which support other floating point representations are allowed.

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 NULLObject. If two Object_References are equal they refer to the same object instance. If an Object_Reference is equal to the special value NULLObject 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.

Enumerated data types

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 which are actually manipulated by the implementation. Enumerated data types are a closed list that do not change based on registration or deprecation. This clause specifies the enumerated data types within this Standard.

Axis_Direction

This data type represents the values of direction the parameter(s) of up SRFT_LOCAL_SPACE_RECTANGULAR_3D and SRFT_LOCAL_SPACE_RECTANGULAR_2D.

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

Polar_Aspect

This data type represents the values of the polar aspect parameter of SRFT_POLAR_STERIOGRAPHIC.

```
Polar_Aspect ::= (
    ASPECT_NORTH,
    ASPECT_SOUTH)
```

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.

EXTENDED_VALID denotes a coordinate that is contained in the extended valid region but not in the valid region.

DEFINED denotes a coordinate that is contained in the CS domain but not in the valid or extended valid regions.

Selection data types

Introduction

Selection data types are similar to enumerated data types but do not form a closed end set of entries. Selection data types are all defined to be of type Short_Integer but with specific meanings attached to each value. The set of selections can be augmented by assigning meanings to additional values. Standardization of additional meanings may occur through registration. Selection data types are otherwise processed in the same manner as enumerated data types.

A language binding may specify symbolic constants for individual selection data type values. In that case, the literal symbol shall be based on the label for that value (if any).

CS_Code

The CS_Code non-object selection data type specifies a coordinate system defined in Clause 5.

```
CS_Code ::= (
    < 1 : // implementation dependent and non-conforming
    1 : // CS_3D_EUCLIDEAN,
    ... : // additional entries of the form i :
           // L, for 2 <= i <= 25, where L is the label denoting the CS also denoted by
code      i,
    26 : // CS_MP_EQUIDISTANT_CYLINDRICAL,
    > 26 : // reserved for registration and future standardization )
```

ORM_Code

The ORM_Code non-object selection data type specifies an ORM defined in Annex E.

```
ORM_Code ::= (
    < 1 : // implementation dependent,
    0 : // INVALID // failure value for API methods that output an ORM_Code
value.
    1 : // ORM_ABSTRACT_2D
    ... : // additional entries of the form C : L, for 2 <= C <= 489, where L is the
label denoting the concept also denoted by code C in Table E.3 through Table E.10.
    482 : // ORM_ZANDERIJ_SURINAME
    > 482 : // reserved for registration )
```

VOS_Code

The VOS_Code non-object selection data type specifies the vertical offset surface used with an ORM realization (see Clause 9).

```
VOS_Code ::= (
    < 1 : // implementation dependent,
    1 : // VOS_EGM96_GEOID,
    2 : // VOS_IGLD_19555,
    3 : // VOS_MSL,
    4 : // VOS_NAVD_1988,
    5 : // VOS_NGVD_1929,
    6 : // VOS_OSGM_2002,
    7 : // VOS_WGS84_GEOID
    > 8 : // reserved for registration )
```

SRFT_Code

The SRFT_Code non-object selection data type specifies an SRF Template (see Clause 8).

```
SRFT_Code ::= (
    < 1      :    // implementation dependent,
    1       :    SRFT_CELESTIOCENTRIC,
    2       :    SRFT_LOCAL_SPACE_RECTANGULAR_3D,
    3       :    SRFT_LOCAL_SPACE_RECTANGULAR_2D,
    4       :    SRFT_CELESTIODETTIC,
    5       :    SRFT_LOCAL_TANGENT_EUCLIDIAN,
    6       :    SRFT_LOCAL_TANGENT_AZIMUTHAL_SPHERICAL,
    8       :    SRFT_AZIMUTHAL,
    9       :    SRFT_LOCAL_CYLINDRICAL_TANGENT_PLANE,
    10      :    SRFT_2D_POLAR,
    11      :    SRFT_CELESTIOMAGNETIC,
    12      :    SRFT_EQUATORIAL_INERTIAL,
    13      :    SRFT_SOLAR_ECLIPTIC,
    14      :    SRFT_SOLAR_EQUITORIAL,
    15      :    SRFT_SOLAR_MAGNETOSPHERIC,
    16      :    SRFT_SOLAR_MAGNETIC,
    17      :    SRFT_HELIOSPHERIC_ARIES_ECLIPTIC,
    18      :    SRFT_HELIOSPHERIC_EARTH_ECLIPTIC,
    19      :    SRFT_HELIOSPHERIC_EARTH_EQUATORIAL,
    20      :    SRFT_MERCATOR,
    21      :    SRFT_OBLIQUE_MERCATOR,
    22      :    SRFT_TRANSVERSE_MERCATOR,
    23      :    SRFT_LAMBERT_CONFORMAL_CONIC,
    24      :    SRFT_POLAR_STEREOGRAPHIC,
    25      :    SRFT_EQUIDISTANT_CYLINDRICAL,
    > 25    :    // reserved for registration )
```

SRF_Code

The SRF_Code non-object selection data type specifies a standardized SRF (see [Clause 8](#)).

```
SRF_Code ::= (
    < 1      :    // implementation dependent,
    0       :    INVALID, // Failure value for API methods that output an
    SRF_Code value.
    1       :    SRF_BRITISH_NATIONAL_GRID,
    2 :    SRF_DELAWARE_SPCS,
    3 :    SRF_GEOCENTRIC_EARTH_1984,
    4 :    SRF_GEODETTIC_AUSTRALIA_1984,
    5 :    SRF_GEODETTIC_AUSTRALIA_1990,
    etc. ...,
    14 :    SRF_MARYLAND_SPCS,
    > 14    :    // reserved for registration )
```

SRFS_Code and SRFS_Member_Code

The SRFS_Code non-object selection data type specifies an SRF Set (see [Clause 8](#)).

```
SRFS_Code ::= (
    < 1      :    // implementation dependent,
    0       :    INVALID, // Failure value for API methods that output an
    SRFS_Code value.
    1       :    SRFS_ALABAMA_SPCS,
    2       :    SRFS_GTRS_GLOBAL_COORDINATE_SYSTEM
    etc. ...,
    7       :    SRFS_UNIVERSAL_TRANSVERSE_MERCATOR,
    8 :    SRFS_WISCONSIN_SPCS,
    > 8      :    // reserved for registration )
```

```
SRFS_Member_Code ::= (
    < 1      :    // implementation dependent,
    0       :    INVALID, // Failure value for API methods that output an
    SRFS_Member_Code value.
    >=1      :    //To be interpreted in the scope of a given SRFS )
```

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 is further defined in the “Error conditions” field of each method specification in Table 0.2 through Table 0.37.

```
Status_Code ::= (
    < 1      :      // implementation_dependent,
    1      :      SUCCESS,      // the operation was performed successfully
    2      :      INVALID_SRF,
    3      :      INVALID_SOURCE_SRF,
    4      :      INVALID_SOURCE_COORDINATE,
    5      :      INVALID_TARGET_COORDINATE
    6      :      OPERATION_UNSUPPORTED,
    7      :      INVALID_SOURCE_DIRECTION,
    8      :      INVALID_CODE,
    9      :      INVALID_INPUT,
    10     :      CREATION_FAILURE,
    11     :      DESTRUCTION_FAILURE,
    12     :      FLOATING_OVERFLOW,
    13     :      FLOATING_UNDERFLOW,
    14     :      FLOATING_POINT_ERROR,
    >15     :      // reserved for registration )
```

Structured data types

Introduction

Non-object data types created as arrays or records whose elements are basic non-object data types are called structured non-object data types. This International Standard specifies two sets of structured non-object data types. The first set of non-object structured data types collect the parameters needed to specify an SRF. The second set is collects the coordinate components that specify a coordinate instance (that is, the value of a spatial position) defined with respect to an SRF.

SRFT parameters

LSR_3D_Parameters

This non-object data type specifies the parameters for SRF SRFT_3D_LOCAL_SPACE_RECTANGULAR (see 0).

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

LSR_2D_Parameters

This non-object data type specifies the parameters that correspond to SRFT_2D_LOCAL_SPACE_RECTANGULAR (see 0).

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

LocalTangentEuclidian_Parameters

This non-object data type specifies the parameters that correspond to SRFT_3D_LOCAL_TANGENT_PLANE (see 0).

```
LocalTangentEuclidian_Parameters ::= {
    orm                                ORM_Code;
    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; }
```

LocalTangent_Parameters

This non-object data type specifies the parameters that correspond to the following SRFs:

SRFT_LOCAL_AZIMUTHAL_SPHERICAL_TANGENT_PLANE, and

SRFT_LOCAL_CYLINDRICAL_TANGENT_PLANE (see 0).

LocalTangent_Parameters ::= {	orm	<u>ORM Code;</u>
	geodetic_longitude	Long_Float;
	geodetic_latitude	Long_Float;
	azimuth	Long_Float;
	height_offset	Long_Float; }

Mercator_Parameters

This non-object data type specifies the parameters that correspond to SRFT_MERCATOR and

SRFT_TRANSVERSE_MERCATOR (see 0).

Mercator_Parameters ::= {	orm	<u>ORM Code:</u>
	standard_longitude	Long_Float;
	standard_latitude	Long_Float;
	central_scale	Long_Float;
	false_easting	Long_Float;
	false_northing	Long_Float; }

Oblique_Mercator_Parameters

This non-object data type specifies the parameters that correspond to `SRFT_OBLIQUE_MERCATOR` (see 0).

Oblique_Mercator_Parameters ::= {	orm	<u>ORM Code</u> ;
	longitude1	Long_Float;
	latitude1	Long_Float;
	longitude2	Long_Float;
	latitude2	Long_Float;
	central_scale	Long_Float;
	false_easting	Long_Float;
	false_northing	Long_Float; }

LCC_Parameters

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

SRFT_LAMBERT_CONFORMAL_CONIC (see 0).

LCC_Parameters ::= {	orm	<u>ORM Code;</u>
	origin_longitude	Long_Float;
	north_parallel_geodetic_latitude	Long_Float;
	south_parallel_geodetic_latitude	Long_Float;
	false_easting	Long_Float;
	false_northing	Long_Float; }

PS_Parameters

This non-object data type specifies the parameters that correspond to SRFT_POLAR_STEREOGRAPHIC (see 0).

PS_Parameters ::= {	orm	<u>ORM Code;</u>
	polar_aspect	<u>Polar Aspect;</u>
	origin_longitude	Long_Float;
	standard_latitude	Long_Float;
	central_scale_factor	Long_Float;
	false_easting	Long_Float;
	false_northing	Long_Float; }

EC_Parameters

This non-object data type specifies the parameters that correspond to `SRFT_EQUIDISTANT_CYLINDRICAL` (see 0).

```
EC_Parameters ::= {
    orm                ORM_Code;
    origin_longitude   Long_Float;
    standard_latitude  Long_Float;
    central_scale_factor Long_Float;
    false_easting      Long_Float;
    false_northing     Long_Float; }
```

Object types

Introduction

SRF objects specify methods that implement the spatial operations specified in [Clause 10](#). The functionality of the API is provided using a class hierarchy with abstract classes that provides a means for providing common methods of subclasses for aid in specification. The abstract classes are not required to be implemented.

The functionality is provided in the tables (see 0) which provide the method name, the semantics, inputs and outputs to the function, and the error conditions of the function. When the functionality is provided within the SRF class hierarchy, the functions are referred to as methods. These methods manipulate internal data (object state) and parameters passed in. The success condition is a nominal behaviour of all methods and is not listed within the error conditions.

EXAMPLE 1: In Table 11.2 [BaseSRF](#), 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 Table 11.2 [BaseSRF](#) abstract method named `GetORMCode` “Outputs the `ORM_Code` of this SRF”, and shows “Inputs: None”.

Language bindings may add additional error conditions and related binding specific mechanisms including memory allocation, the passing of inputs and outputs, and the presentation of method status. Language bindings shall specify these mechanisms, since this part of 18026 does not restrict such mechanisms. Under an error condition, output values are undefined, but a language binding may specify the output values. When several error conditions apply to a method invocation, the first error condition detected by an implementation shall be presented as the method status.

A language binding mechanism for presentation of method status shall support the association of a unique error `Status_Code` (11.2.5.5).

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 which returns the corresponding `Status_Code` would satisfy this requirement.

Class Specification Format

Class types are specified in tables in [Table 0.2](#) through [Table 0.37](#) with the following fields:

Field label		Field description
Class		Name of the private object class
Description		The corresponding SRM concept
Superclass		Specification of inherited functionality
(Abstract) method	Name	Name of the method
	Semantics	Specification of the functionality
	Inputs	Specification(s) of the input parameters, or “None”. The state of the invoking object is implicitly an input and is not listed in this field.
	Outputs	Specification(s) of the output parameters, or “None”.
	Error conditions	A list of <code>Status_Code</code> values. Each listed value specifies the condition for which it is applicable. Common error conditions (see below) are not listed in this field.

The method field of an Abstract class is labelled “Abstract method”. The method field of a concrete class is labelled “Method”. A subclass inherits all the methods of its superclass including methods that the superclass has inherited.

The method Error conditions field does not separately list the following common error conditions.

- f. 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.
- g. FLOATING_OVERFLOW if a floating point overflow error occurred in the performance of the method.
- h. FLOATING_UNDERFLOW if a floating point underflow error occurred in the performance of the method.
- i. FLOATING_POINT_ERROR if floating point error (other than overflow or underflow) occurred in the performance of the method.
- j. INVALID_INPUT if a Long_Float input is positive or negative infinity or a NAN value. If additional conditions apply, they are listed with this Status_Code in the (Abstract) Method Error conditions field.

LifeCycleObject

The LifeCycleObject class is the most basic abstract class from which all other classes inherit. Other abstract classes are defined in 0.

Table 0.2 — LifeCycleObject

Class	LifeCycleObject	
Description	This is the most basic abstract class from which all other classes inherit. An object derived from LifeCycleObject is invalid until the Create method is successfully invoked. A valid object is invalid after the Destroy method is invoked.	
Superclass	None	
Abstract method	Name	Create
	Semantics	This method creates an instance of an object. An implementation may perform memory allocation and/or object initialization as part of this method.
	Inputs	Specific inputs are specified in concrete classes which are directly or indirectly sub-classed from this class type.
	Outputs	objRef: <u>Object Reference</u>
	Error conditions	2. CREATION_FAILED if the object instance could not be created.
Abstract method	Name	Destroy
	Semantics	This method destroys an instance of an object. An implementation may perform memory deallocation and/or object finalization as part of this method.
	Inputs	objRef: <u>Object Reference</u>
	Outputs	None
	Error conditions	3. DESTRUCTION_FAILURE if the object was not successfully created through the API.

Private Objects

Introduction

Private object data types are concrete classes that represent objects whose methods and attributes are not exposed to the API user. Instances of these objects may be created and destroyed by methods on other objects specified in the API.

Therefore, their object references may be passed to and returned from methods. This allows an implementation of the API to store data that must be maintained for these object data types in whatever form is convenient for the implementation.

Coordinate3D

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

Table 0.3 —Coordinate3D

Class	Coordinate3D
Description	A coordinate of CS type 3D (see 5.2.2).
Superclass	LifeCycleObject

Coordinate2D

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

Table 0.4 — Coordinate2D

Class	Coordinate2D
Description	A coordinate of CS type 2D (see 5.2.2).
Superclass	LifeCycleObject

CoordinateSurf

The CoordinateSurf class represents a coordinate of CS type Surface. It specifies no additional public methods.

Table 0.5 —CoordinateSurf

Class	CoordinateSurf
Description	A surface coordinate of CS type surface (see 5.2.4.2).
Superclass	LifeCycleObject

Direction

The Direction class represents a direction in a 3D object-space. It specifies no additional public methods.

Table 0.6 — Direction

Class	Direction
Description	A direction (see 10.5.1).
Superclass	LifeCycleObject

Abstract classes

BaseSRF

This is the base class for all SRFT classes.

Table 0.7 — BaseSRF

Class	BaseSRF	
Description	An abstract class specifying the common methods of all SRFT classes	
Superclass	<u>LifeCycleObject</u>	
Abstract method	Name	GetORMCode
	Semantics	Outputs the ORM_Code of this SRF.
	Inputs	None.
	Outputs	orm
	Error conditions	No additional error conditions. (See 11.3.x a.)
Abstract method	Name	GetCodes
	Semantics	1. Outputs the SRFT_Code of this SRF3D instance. 2. If created by the CreateStandardSRF function, outputs a valid SRF_Code (otherwise 0). See 11.3.3.x 3. If created by an SRF set Class, outputs a valid SRFS_Code (otherwise 0) and a valid SRF_Member_Code (otherwise 0). See 11.3.3.y
	Inputs	None.
	Outputs	srft: <u>11.2.5.4 SRFT, SRF, and SRFS and SRFS member codes</u> srf: <u>SRF_Code</u> srfs: <u>SRFS_Code</u> srfs_member: <u>SRFS_Member_Code</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)
Abstract method	Name	GetCSCode
	Semantics	Outputs the CS_Code of this SRF.
	Inputs	None.
	Outputs	cs_code <u>CS_Code</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

BaseSRF2D

This is the base class for all 2D SRFs.

Table 0.8 — BaseSRF2D

Class	BaseSRF2D
Description	An abstract class representing the common elements of SRFTs that have a coordinate system of type 2D.
Superclass	<u>BaseSRF</u>

Class	BaseSRF2D	
Abstract method	Name	CreateCoordinate2D
	Semantics	This method accepts the two ordered coordinate components and for a specific SRF creates a 2D coordinate instances initialised with the values passed in.
	Inputs	first_value Long_Float second_value Long_Float
	Outputs	coordinate <u>Coordinate2D</u>
	Error conditions	1. INVALID_SOURCE_COORD if the coordinate values are not in the domain of the coordinate system generating function as specified in 5.4.
Abstract method	Name	GetCoordinate2DValues
	Semantics	This method retrieves the two ordered coordinate components of a 2D coordinate instance previously initialised through the API
	Inputs	coordinate <u>Coordinate2D</u>
	Outputs	first_value Long_Float second_value Long_Float
	Error conditions	3. INVALID_SOURCE_COORD if the coordinate was not a 2D coordinate, not initialised through the AP, or is not in the CS domain of this SRF.
Abstract method	Name	ChangeCoordinate2DSRF
	Semantics	This method changes the SRF representation of the spatial position specified by the input Coordinate2D, source_coordinate, from this SRF to a Coordinate2D representation in the target SRF, target_srf, in accordance with sub-clause 10.4.1.
	Inputs	source_srf: BaseSRF2D source_coordinate: <u>Coordinate2D</u>
	Outputs	target_coordinate: <u>Coordinate2D</u>
	Error conditions	6. INVALID_SOURCE_COORDINATE if source_coordinate is not in this SRF. 7. INVALID_SOURCE_SRF, if source_srf was not successfully initialised through the API or is invalid. 8. OPERATION_UNSUPPORTED, If the source_srf is an SRF for a different spatial object. 9. INVALID_TARGET_COORDINATE if the spatial position is not in the CS domain of this SRF.
Abstract method	Name	EuclideanDistance
	Semantics	Outputs the Euclidean distance (in metres) between the spatial points represented by Coordinates2D source_coordinate and target_coordinate.

Class	BaseSRF2D	
	Inputs	source_coordinate <u>Coordinate2D</u> target_coordinate <u>Coordinate2D</u>
	Outputs	distance Long_Float
	Error conditions	5. INVALID_SOURCE_COORDINATE if source_coordinate is not in the CS domain of this SRF. 6. INVALID_TARGET_COORDINATE, if target_coordinate is not in the CS domain of this SRF.

BaseSRF3D

This is the base class for 3D SRFs.

Table 0.9 — BaseSRF3D

Class	BaseSRF3D	
Description	An abstract class representing the common elements of SRFTs that have a coordinate system of type 3D.	
Superclass	<u>BaseSRF</u>	
Abstract method	Name	CreateCoordinate3D
	Semantics	This method creates a <u>Coordinate3D</u> for this SRF from three ordered coordinate component values.
	Inputs	First_coordinate_component: Long_Float second_coordinate_component: Long_Float third_coordinate_component: Long_Float
	Outputs	coordinate: <u>Coordinate3D</u>
	Error conditions	2. INVALID_SOURCE_COORD if the coordinate values are not in the domain of the coordinate system generating function as specified in 5.4.

Class	BaseSRF3D	
Abstract method	Name	CreateDirection
	Semantics	This method accepts a 3D coordinate and the three direction components and a reference coordinate for -this SRF and creates a direction instance initialised with the values passed in. The direction vector is normalized.
	Input	reference_coordinate: <u>Coordinate3D</u> first_direction_component: Long_Float second_direction_component: Long_Float third_direction_component: Long_Float
	Output	direction_out <u>Direction</u>
	Error conditions	4. INVALID_SOURCE_COORD if reference_coordinate is not in the CS domain of this SRF or is invalid. 5. INVALID_SOURCE_DIRECTION If the direction components are all zero.
Abstract method	Name	GetCoordinate3Dvalues
	Semantics	This method retrieves the three ordered coordinate components of a 3D coordinate instance previously initialised through the API
	Inputs	coordinate: <u>Coordinate3D</u>
	Outputs	first_coordinate_component: Long_Float second_coordinate_component: Long_Float third_coordinate_component: Long_Float
	Error conditions	3. INVALID_SOURCE_COORD if the coordinate was not a 3D coordinate, not initialised through the API, or is not in the CS domain for this SRF.
Abstract method	Name	GetDirectionValues
	Semantic	This method retrieves the reference coordinate and the three components of a direction instance previously initialised through the API.
	Input	direction_in <u>Direction</u>
	Output	reference_coordinate: <u>Coordinate3D</u> first_value Long_Float second_value Long_Float third_value Long_Float
	Error Conditions	3. INVALID_SOURCE_DIRECTION if the reference coordinate was not a 3D coordinate for this SRF or not initialised through the API.

Class	BaseSRF3D	
Abstract method	Name	ChangeCoordinate3DSRF
	Semantic	<p>This method changes the SRF representation of the spatial position specified by the input <u>Coordinate3D</u>, source_coordinate, from the source SRF, source_srf, to a <u>Coordinate3D</u> representation in this SRF in accordance with sub-clause 10.4.1.</p> <p>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.</p>
	Inputs	source_srf: Base3DSRF source_coordinate: <u>Coordinate3D</u>
	Outputs	target_coordinate: <u>Coordinate3D</u> region: <u>Coordinate Valid Region</u>
	Error conditions	7. INVALID_SOURCE_COORDINATE if source_coordinate is not in the CS domain of the SRF specified by source_srf. 8. INVALID_SOURCE_SRF, if source_srf was not successfully initialised through the API or is invalid. 9. OPERATION_UNSUPPORTED, If the source_srf is an SRF for a different spatial object. 10. INVALID_TARGET_COORDINATE if the spatial position is not in the CS domain of this SRF.

Class	BaseSRF3D	
Abstract method	Name	ChangeDirectionSRF
	Semantics	This method changes the SRF representation of the direction by the input Direction, source_direction, from the source SRF, source_srf, to a Direction representation in this SRF in accordance with sub-clause 10.5.3. 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.
	Input	source_srf: Base3DSRF source_direction: <u>Direction</u>
	Output	target_direction: <u>Direction</u> ref_coord_region <u>Coordinate_Valid_Region</u>
	Error conditions	7. INVALID_SOURCE_DIRECTION If the reference coordinate of the Direction is invalid or if the direction components are all zero. 8. INVALID_SOURCE_SRF, if source_srf was not successfully initialised through the API or is invalid. 9. OPERATION_UNSUPPORTED, if the source_srf is an SRF for a different spatial object. 10. INVALID_TARGET_COORDINATE if the reference location of the direction is not in the CS domain of this SRF.
Abstract method	Name	EuclideanDistance
	Semantics	Outputs the Euclidean distance (in metres) between the spatial points represented by <u>Coordinate3D</u> source_coordinate and target_coordinate.
	Inputs	source_coordinate <u>Coordinate3D</u> target_coordinate <u>Coordinate3D</u>
	Outputs	distance Long_Float
	Error conditions	1. INVALID_SOURCE_COORDINATE if source_coordinate is not in the CS domain of this SRF. 2. INVALID_TARGET_COORDINATE, if target_coordinate is not in the CS domain of this SRF.

Note: The method ChangeCoordinate3DSRF requires an implementation to operate on a source coordinate from an unknown SRF object derived from BaseSRF3D. The clause 10 methodology for this operation supports this kind of extensibility. For example, if an implementation design requires each concrete (direct or indirect) subclass of BaseSRF3D to implement (private) methods for the generating function of the SRFT and its inverse, and the transformation of the ORM to the reference ORM and its inverse, and if these methods are accessible (with a standard syntax) to all BaseSRF3D derived classes, then one generic way of changing the source coordinate from an otherwise unknown BaseSRF3D derived source SRF is to have the ChangeCoordinate3DSRF method of the target SRF: (1) apply the source SRF generating function to the source coordinate, (2) apply the source ORM

transformation to the result, (3) then apply the target SRF inverse ORM transformation, and (4) finally apply the target SRF inverse generating function to produce the target coordinate. This computational scheme is presented for illustrative purpose only, and does not take into account error checking and various computational short cuts and efficiencies that an actual design would consider. (Some of these efficiencies are presented in Clause 10). This remark is also applicable to the ChangeDirectionSRF method.

BaseSRFwithTangentPlaneSurface

This is the base class for the LocalTangentPlane, LocalAzimuthalSphericalTangentPlane, and LocalCylindricalTangentPlane SRF classes. It is based on the BaseSRF3D class. It adds methods for the surface CS that is induced on the third coordinate surface for zero which is a plane that is tangent to the oblate spheroid RD of the ORM of the SRF.

Class	BaseSRFwithTangentPlaneSurface	
Description	This class is a 3D SRF for which the third coordinate surface for zero is a tangent plane to the oblate spheroid RD of the ORM of the SRF. A surface CS is induced on the third coordinate surface for zero.	
Superclass	<u>BaseSRF3D</u>	
Abstract method	Name	CreateCoordinateSurf
	Semantics	Creates a surface coordinate on the tangent plane surface. The output is invalid if the method does not succeed.
	Inputs	first_component Long_Float seond_component Long_Float
	Outputs	new_coordinate <u>CoordinateSurf</u>
	Error conditions	4. INVALID_SOURCE_COORDINATE if the input values are not in the domain of the induced surface CS generating function as specified in 5.4.

Class	BaseSRFwithTangentPlaneSurface	
Abstract method	Name	GetCoordinateSurfValues
	Semantics	Retrieves the component values of a surface coordinate on the tangent plane surface. The output is invalid if the method does not succeed.
	Inputs	coordinate <u>CoordinateSurf</u>
	Outputs	first_component Long_Float seond_component Long_Float
	Error conditions	4. INVALID_SOURCE_COORDINATE if coordinate is not a valid surface coordinate in this SRF.
Abstract method	Name	AssociateCoordinateSurf
	Semantics	Creates the CoordinateSurf associated with a <u>Coordinate3D</u> by setting the third coordinate component to zero and then converting that coordinate to its Surface CS representation (Truncate to surface).
	Inputs	coordinate_3D <u>Coordinate3D</u>
	Outputs	on_surface_coordinate: <u>CoordinateSurf</u>
	Error conditions	4. INVALID_SOURCE_COORDINATE if coordinate_3D is not a valid 3D coordinate in this SRF.
Abstract method	Name	PromoteSurfaceCoordinate
	Semantics	Creates a <u>Coordinate3D</u> representing the same location as specified by on_surface_coordinate (Promote surface coordinate to 3D coordinate).
	Inputs	surface_coordinate: <u>CoordinateSurf</u>
	Outputs	coordinate_on_the_surface: <u>Coordinate3D</u>
	Error conditions	4. INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate in this SRF.
Abstract method	Name	EuclideanDistance
	Semantics	Outputs the Euclidean distance (in metres) between the spatial points represented by <u>CoordinateSurfs</u> source_coordinate and target_coordinate.
	Inputs	source_coordinate <u>CoordinateSurf</u> target_coordinate <u>CoordinateSurf</u>
	Outputs	distance Long_Float

Class	BaseSRFwithTangentPlaneSurface	
	Error conditions	4. INVALID_SOURCE_COORDINATE if source_coordinate is not in the CS domain of this SRF. 5. INVALID_TARGET_COORDINATE, if target_coordinate is not in the CS domain of this SRF.

BaseSRFwithEllipsoidalHeight

This is the base class for the celestiodetic SRF class and BaseMapProjection Class. It is based on the BaseSRF3D class. It adds methods for the surface CS that is induced on the zero ellipsoidal height surface. It also adds a method to create a local tangent plane SRF on a point on the zero ellipsoidal height surface.

Table 0.10 — BaseSRFwithEllipsoidalHeight

Class	BaseSRFwithEllipsoidalHeight	
Description	This class is a 3D SRF with ellipsoidal height as third coordinate component. An abstract class representing the common elements of SRFTs that have an ORM with an oblate spheroid RD and a coordinate system of type 3D with ellipsoidal height (based on the RD) as the third coordinate component. A surface CS is induced on the third coordinate surface for zero.	
Superclass	<u>BaseSRF3D</u>	
Abstract method	Name	CreateSurfaceCoordinate
	Semantics	Creates a surface coordinate on the ORM surface. The output is invalid if the method does not succeed.
	Inputs	first_component Long_Float seond_component Long_Float
	Outputs	new_coordinate <u>CoordinateSurf</u>
	Error conditions	1. INVALID_SOURCE_COORDINATE if the input values are not in the domain of the induced surface CS generating function as specified in 5.4.

Class	BaseSRFwithEllipsoidalHeight	
Abstract method		GetSurfaceCoordinateValues
	Semantics	Retrieves the component values of a surface coordinate on the ORM surface. The output is invalid if the method does not succeed.
	Inputs	coordinate <u>CoordinateSurf</u>
	Outputs	first_component Long_Float seond_component Long_Float
	Error conditions	1. INVALID_SOURCE_COORDINATE if coordinate is not a valid surface coordinate in this SRF.
Abstract method	Name	AssociateSurfaceCoordinate
	Semantics	Creates the <u>CoordinateSurf</u> associated with a <u>Coordinate3D</u> by setting the third coordinate component to zero and then converting that coordinate to its Surface CS representation (Truncate to surface).
	Inputs	coordinate_3D <u>Coordinate3D</u>
	Outputs	on_surface_coordinate: <u>CoordinateSurf</u>
	Error conditions	1. INVALID_SOURCE_COORDINATE if coordinate_3D is not a valid 3D coordinate in this SRF.
Abstract method		PromoteSurfaceCoordinate
	Semantics	Creates a <u>Coordinate3D</u> representing the same location as specified by on_surface_coordinate (Promote Surface to 3D).
	Inputs	surface_coordinate: <u>CoordinateSurf</u>
	Outputs	coordinate_on_the_surface: <u>Coordinate3D</u>
	Error conditions	1. INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate in this SRF.
Abstract method	Name	CreateLocalTangentEuclideanSRF
	Semantics	Creates a LocalTangentEuclidian SRF with natural origin at the input <u>CoordinateSurf</u> . 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.
	Inputs	surface_coordinate: <u>CoordinateSurf</u> azimuth: Long_Float false_x_origin: Long_Float false_y_origin: Long_Float offset_height: Long_Float
	Outputs	localtangentEuclidian_srf: LocalTangentEuclidian

Class	BaseSRFwithEllipsoidalHeight	
	Error conditions	3. INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate this SRF.
Abstract method	Name	VerticalSeparationOffset
	Semantics	Outputs the vertical separation offset (see 9.2.4) at the input surface coordinate between the oblate spheroid RD of this SRF and the specified VOS.
	Inputs	vos_code: <u>VOS_Code</u> , surface_coordinate: <u>CoordinateSurf</u>
	Outputs	separation: Long_Float
	Error conditions	4. INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate for this SRF 5. INVALID_CODE if the vos is not a valid VOS_CODE 6. UNSUPPORTED_OPERATION if the VOS is not defined for the oblate spheroid RD of this SRF, or if the offset vertical separation is undefined at the surface location.
Abstract method	Name	GeodesicDistance
	Semantics	Outputs the Geodesic distance (in metres) between source to target positions on the Surface of the oblate spheroid RD.
	Inputs	source_coordinate: <u>CoordinateSurf</u> target_coordinate: <u>CoordinateSurf</u>
	Outputs	distance Long_Float
	Error conditions	1. INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate this SRF. 2. INVALID_TARGET_COORDINATE if target_coordinate is not a valid surface coordinate for this SRF.
Abstract method	Name	EuclideanDistance
	Semantics	Outputs the Euclidean distance (in metres) between the spatial points represented by <u>CoordinateSurfs</u> source_coordinate and target_coordinate.
	Inputs	source_coordinate <u>CoordinateSurf</u> target_coordinate <u>CoordinateSurf</u>
	Outputs	distance Long_Float

Class	BaseSRFwithEllipsoidalHeight	
	Error conditions	1. INVALID_SOURCE_COORDINATE if source_coordinate is not in the CS domain of this SRF. 2. INVALID_TARGET_COORDINATE, if target_coordinate is not in the CS domain of this SRF.

BaseMapProjection

Table 0.11 — BaseMapProjection

Class	BaseMapProjection	
Description	An abstract class representing the common elements of SRFTs that have a map projections coordinate system of type 3D.	
Superclass	<u>BaseSRFwithEllipsoidalHeight</u>	
Abstract method	Name	ConvergenceOfTheMeridian
	Semantics	Outputs the Convergence of the Meridian in radians at a position on the Surface of the oblate spheroid RD.
	Inputs	surface_coordinate <u>CoordinateSurf</u> ;
	Outputs	gamma Long_Float
	Error conditions	1. INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate this SRF.
Abstract method	Name	PointScale
	Semantics	Outputs the point scale at a position on the Surface of the oblate spheroid RD.
	Inputs	surface_coordinate <u>CoordinateSurf</u> ;
	Outputs	scale Long_Float
	Error conditions	1. INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate this SRF.
Abstract method	Name	MapAzimuth
	Semantics	Outputs the map azimuth in radians at from a source to target positions on the Surface of the oblate spheroid RD.
	Inputs	source_coordinate <u>CoordinateSurf</u> ; target_coordinate <u>CoordinateSurf</u> ;
	Outputs	azimuth Long_Float
	Error conditions	1. INVALID_SOURCE_COORDINATE if surface_coordinate is not a valid surface coordinate this SRF. 2. INVALID_TARGET_COORDINATE, if target_coordinate is not a valid surface coordinate in this SRF.

SRF concrete subclasses of BaseSRF2D

LocalSpaceRectangular2D

Table 0.12 — LocalSpaceRectangular2D

Class	LocalSpaceRectangular2D	
Description	An instance of this class corresponds to an instance of SRFT_2D_LOCAL_SPACE_RECTANGULAR	
Superclass	<u>BaseSRF2D</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a 2D local space rectangular SRF corresponding to the input parameter.
	Inputs	parameters: <u>LSR_2D_Parameters</u>
	Outputs	new_srf: LocalSpaceRectangular2D
	Error Conditions	2. INVALID_INPUT if input parameter is not valid for this SRF.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF parameters.
	Inputs	None
	Outputs	parameters: <u>LSR_2D_Parameters</u>
	Error conditions	No additional error conditions (see 11.3.x a.)

Azimuthal

Table 0.13 — Azimuthal

Class	Azimuthal	
Description	An instance of this class corresponds to an instance of SRFT_2D_AZIMUTHAL.	
Superclass	<u>BaseSRF2D</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates an Azimuthal 2D SRF corresponding to the input parameter.
	Inputs	Orm:
	Outputs	New_srf: Azimuthal
	Error Conditions	1. INVALID_CODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF ORM_Code parameter.
	Inputs	None
	Outputs	orm_code: <u>ORM_Code</u>
	Error conditions	No additional error conditions (see 11.3.x a.)

Table 0.14 — Polar

Class	Polar	
Description	An instance of this class corresponds to an instance of SRFT_2D_POLAR.	
Superclass	<u>BaseSRF2D</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a Polar 2D SRF corresponding to the input ORM_Code parameter.
	Inputs	Orm_code: <u>ORM Code</u>
	Outputs	New_srf: Polar
	Error Conditions	2. INVALID_CODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF ORM_Code parameter.
	Inputs	None
	Outputs	orm_code: <u>ORM Code</u>
	Error conditions	No additional error conditions (see 11.3.x a.)

SRF concrete subclasses of BaseSRF3D

LocalSpaceRectangular3D

Table 0.15 — LocalSpaceRectangular3D

Class	LocalSpaceRectangular3D	
Description	An instance of this class corresponds to an instance of SRFT_3D_LOCAL_SPACE_RECTANGULAR	
Superclass	<u>BaseSRF3D</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a 3D local space rectangular SRF corresponding to the input parameters.
	Inputs	parameters: <u>LSR_3D_Parameters</u>
	Outputs	new_srf: LocalSpaceRectangular3D
	Error conditions	2. INVALID_INPUT if input parameters are not valid for this SRF.

Class	LocalSpaceRectangular3D	
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF parameters.
	Inputs	None
	Outputs	parameters: <u>LSR_3D_Parameters</u>
	Error conditions	No additional error conditions (see 11.3.x a.)

Celestiocentric

Table 0.16 — Celestiocentric

Class	Celestiocentric	
Description	An instance of this class corresponds to an instance of SRFT_CELESTIOCENTRIC	
Superclass	<u>BaseSRF3D</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a celestiocentric SRF corresponding to the input parameter.
	Inputs	orm_code: <u>ORM_Code</u>
	Outputs	new_srf: Celestiocentric
	Error conditions	2. INVALID_CODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF ORM_Code parameter.
	Inputs	None.
	Outputs	orm_code: <u>ORM_Code</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

CelestioMagnetic

Table 0.17 — CelestioMagnetic

Class	CelestioMagnetic	
Description	An instance of this class corresponds to an instance of SRFT_CELESTIOMAGNETIC.	

Class	CelestioMagnetic	
Superclass	<u>BaseSRF3D</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a CelestioMagnetic SRF corresponding to the input parameter.
	Inputs	orm_code: <u>ORM_Code</u>
	Outputs	new_srf: CelestioMagnetic
	Error conditions	2. INVALID_CODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF ORM_Code parameter.
	Inputs	None
	Outputs	orm_code: <u>ORM_Code</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

EquatorialInertial

Table 0.18 — EquatorialInertial

Class	EquatorialInertial	
Description	An instance of this class corresponds to an instance of SRFT_EQUATORIAL_INERTIAL.	
Superclass	<u>BaseSRF3D</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a EquatorialInertial SRF corresponding to the input parameter.
	Inputs	orm_code: <u>ORM_Code</u>
	Outputs	new_srf: EquatorialInertial
	Error conditions	1. INVALID_CODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF ORM_Code parameter.
	Inputs	None
	Outputs	orm_code: <u>ORM_Code</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

SolarEcliptic

Table 0.19 — SolarEcliptic

Class	SolarEcliptic	
Description	An instance of this class corresponds to an instance of SRFT_SOLAR_ECLIPTIC	
Superclass	<u>BaseSRF3D</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a SolarEcliptic SRF corresponding to the input parameter.
	Inputs	orm_code: <u>ORM_Code</u>
	Outputs	new_srf: SolarEcliptic
	Error conditions	1. INVALID_CODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF ORM_Code parameter.
	Inputs	None
	Outputs	orm_code: <u>ORM_Code</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

SolarEquatorial

Table 0.20 — SolarEquatorial

Class	SolarEquatorial	
Description	An instance of this class corresponds to an instance of SRFT_SOLAR_EQUATORIAL	
Superclass	<u>BaseSRF3D</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a SolarEquatorial SRF corresponding to the input parameter.
	Inputs	orm_code: <u>ORM_Code</u>
	Outputs	new_srf: SolarEquatorial
	Error conditions	1. INVALID_CODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF ORM_Code parameter.
	Inputs	None
	Outputs	orm_code: <u>ORM_Code</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

SolarMagnetospheric

Table 0.21 — SolarMagnetospheric

Class	SolarMagnetospheric	
Description	An instance of this class corresponds to an instance of SRFT_SOLAR_MAGNETOSPHERIC.	
Superclass	<u>BaseSRF3D</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a SolarMagnetospheric SRF corresponding to the input parameter.
	Inputs	orm_code: <u>ORM_Code</u>
	Outputs	new_srf: SolarMagnetospheric
	Error conditions	1. INVALID_CODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF ORM_Code parameter.
	Inputs	None
	Outputs	orm_code: <u>ORM_Code</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

SolarMagnetic

Table 0.22 — SolarMagnetic

Class	SolarMagnetic	
Description	An instance of this class corresponds to an instance of SRFT_SOLAR_MAGNETIC.	
Superclass	<u>BaseSRF3D</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a SolarMagnetic SRF corresponding to the input parameter.
	Inputs	orm_code: <u>ORM_Code</u>
	Outputs	new_srf: SolarMagnetic
	Error conditions	1. INVALID_CODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.

Class	SolarMagnetic	
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF ORM_Code parameter.
	Inputs	None
	Outputs	orm_code: <u>ORM_Code</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

HeliosphericAriesEcliptic

Table 0.23 — HeliosphericAriesEcliptic

Class	HeliosphericAriesEcliptic	
Description	An instance of this class corresponds to an instance of SRFT_HELIOSPHERIC_ARIES_ECLIPTIC	
Superclass	<u>BaseSRF3D</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a HeliosphericAriesEcliptic SRF corresponding to the input parameter.
	Inputs	orm_code: <u>ORM_Code</u>
	Outputs	new_srf: HeliosphericAriesEcliptic
	Error conditions	1. INVALID_CODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF ORM_Code parameter.
	Inputs	None
	Outputs	orm_code: <u>ORM_Code</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

HeliosphericEarthEcliptic

Table 0.24 — HeliosphericEarthEcliptic

Class	HeliosphericEarthEcliptic	
Description	An instance of this class corresponds to an instance of SRFT_HELIOSPHERIC_EARTH_ECLIPTIC	

Class	HeliosphericEarthEcliptic	
Superclass	<u>BaseSRF3D</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a HeliosphericEarthEcliptic SRF corresponding to the input parameter.
	Inputs	orm_code: <u>ORM_Code</u>
	Outputs	new_srf: HeliosphericEarthEcliptic
	Error conditions	1. INVALID_CODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF ORM_Code parameter.
	Inputs	None
	Outputs	orm_code: <u>ORM_Code</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

HeliosphericEarthEquatorial

Table 0.25 — HeliosphericEarthEquatorial

Class	HeliosphericEarthEquatorial	
Description	An instance of this class corresponds to an instance of SRFT_HELIOSPHERIC_EARTH_EQUATORIAL	
Superclass	<u>BaseSRF3D</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a HeliosphericEarthEquatorial SRF corresponding to the input parameter.
	Inputs	orm_code: <u>ORM_Code</u>
	Outputs	new_srf: HeliosphericEarthEquatorial
	Error conditions	1. INVALID_CODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF ORM_Code parameter.
	Inputs	None
	Outputs	orm_code: <u>ORM_Code</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

SRF concrete subclasses of BaseSRFwithTangentPlaneSurface

LocalTangentEuclidean

Table 0.26 — LocalTangentEuclidean

Class	LocalTangentEuclidean	
Description	An instance of this class corresponds to an instance of SRFT_3D_LOCAL_TANGENT_PLANE	
Superclass	<u>BaseSRFwithTangentPlaneSurface</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a LocalTangentEuclidean SRF corresponding to the input parameter.
	Inputs	parameters: <u>LocalTangentEuclidian_Parameters</u>
	Outputs	new_srf: LocalTangentEuclidean
	Error conditions	1. INVALID_INPUT if input parameters are not valid for this SRF.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF parameters.
	Inputs	None
	Outputs	parameters: <u>LocalTangentEuclidian_Parameter_s</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

LocalTangentAzimuthalSpherical

Table 0.27 — LocalTangentAzimuthalSpherical

Class	LocalTangentAzimuthalSpherical	
Description	An instance of this class corresponds to an instance of SRFT_LOCAL_TANGENT_AZIMUTHAL_SPHERICAL	
Superclass	<u>BaseSRFwithTangentPlaneSurface</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a LocalTangentEuclidean SRF corresponding to the input parameter.
	Inputs	parameters: <u>LocalTangent_Parameters</u>
	Outputs	new_srf: LocalTangentAzimuthalSpherical
	Error conditions	1. INVALID_INPUT if input parameters are not valid for this SRF.

Class	LocalTangentAzimuthalSpherical	
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF parameters.
	Inputs	None
	Outputs	parameters: <u>LocalTangent_Parameters</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

LocalTangentCylindrical

Table 0.28 — LocalTangentCylindrical

Class	LocalTangentCylindrical	
Description	An instance of this class corresponds to an instance of SRFT_3D_LOCAL_TANGENT_CYLINDRICAL	
Superclass	<u>BaseSRFwithTangentPlaneSurface</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a LocalCylindricalTangentPlane SRF corresponding to the input parameter.
	Inputs	parameters: <u>LocalTangent_Parameters</u>
	Outputs	new_srf: <u>LocalTangentCylindrical</u>
	Error conditions	1. INVALID_INPUT if input parameters are not valid for this SRF.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF parameters.
	Inputs	None
	Outputs	parameters: <u>LocalTangent_Parameters</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

SRF concrete subclasses of BaseSRFwithEllipsoidalHeight

Celestiodetic

Table 0.29 — Celestiodetic

Class	Celestiodetic	
Description	An instance of this class corresponds to an instance of SRFT_CELESTIODETTIC	
Superclass	<u>BaseSRFwithEllipsoidalHeight</u>	

Class	Celestiodetic	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a Celestiodetic SRF corresponding to the input parameter.
	Inputs	orm_code: <u>ORM_Code</u>
	Outputs	new_srf: Celestiodetic
	Error conditions	1. INVALID_CODE if orm is not a valid ORM_Code or the corresponding ORM is not valid for this SRFT.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF ORM_Code parameter.
	Inputs	None
	Outputs	orm_code: <u>ORM_Code</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

SRF concrete subclasses of BaseMapProjection

Table 0.30 — Mercator

Class	Mercator	
Description	An instance of this class corresponds to an instance of SRFT_MERCATOR	
Superclass	<u>BaseMapProjection</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a Mercator SRF corresponding to the input parameter.
	Inputs	parameters: <u>Mercator_Parameters</u>
	Outputs	new_srf: Mercator
	Error conditions	1. INVALID_INPUT if parameters are not valid for this SRF.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF parameters.
	Inputs	None
	Outputs	parameters: <u>Mercator_Parameters</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

ObliqueMercator

Table 0.31 — ObliqueMercator

Class	ObliqueMercator	
Description	An instance of this class corresponds to an instance of SRFT_OBLIQUE_MERCATOR	
Superclass	<u>BaseMapProjection</u>	

Class	ObliqueMercator	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a ObliqueMercator SRF corresponding to the input parameters.
	Inputs	parameters: <u>Oblique Mercator Parameters</u>
	Outputs	new_srf: ObliqueMercator
	Error conditions	1. INVALID_INPUT if parameters are not valid for this SRF.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF parameters.
	Inputs	None
	Outputs	parameters: <u>Oblique Mercator Parameters</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

TransverseMercator

Table 0.32 — TransverseMercator

Class	TransverseMercator	
Description	An instance of this class corresponds to an instance of SRFT_TRANSVERSE_MERCATOR	
Superclass	<u>BaseMapProjection</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a TransverseMercator SRF corresponding to the input parameters.
	Inputs	parameters: <u>Mercator Parameters</u>
	Outputs	new_srf: TransverseMercator
	Error conditions	1. INVALID_INPUT if parameters are not valid for this SRF.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF parameters.
	Inputs	None
	Outputs	parameters: <u>Mercator Parameters</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

LambertConformalConic

Table 0.33 — LambertConformalConic

Class	LambertConformalConic	
Description	An instance of this class corresponds to an instance of SRFT_LAMBERT_CONFORMAL_CONIC	
Superclass	<u>BaseMapProjection</u>	

Class	LambertConformalConic	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a LambertConformalConic SRF corresponding to the input parameters.
	Inputs	parameters: <u>LCC Parameters</u>
	Outputs	new_srf: LambertConformalConic
	Error conditions	1. INVALID_INPUT if parameters are not valid for this SRF.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF parameters.
	Inputs	None
	Outputs	parameters: <u>LCC Parameters</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

PolarStereographic

Table 0.34 — PolarStereographic

Class	PolarStereographic	
Description	An instance of this class corresponds to an instance of SRFT_POLAR_STEREOGRAPHIC	
Superclass	<u>BaseMapProjection</u>	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a PolarStereographic SRF corresponding to the input parameters.
	Inputs	parameters: <u>PS Parameters</u>
	Outputs	new_srf: PolarStereographic
	Error conditions	1. INVALID_INPUT if parameters are not valid for this SRF.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF parameters.
	Inputs	None
	Outputs	parameters: <u>PS Parameters</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

Equidistant Cylindrical

Table 0.35 — Equidistant Cylindrical

Class	EquidistantCylindrical	
Description	An instance of this class corresponds to an instance of SRFT_EQUIDISTANT_CYLINDRICAL	
Superclass	<u>BaseMapProjection</u>	

Class	EquidistantCylindrical	
Method	Name	Create
	Semantics	Overrides the Create method on the superclass <u>LifeCycleObject</u> . Creates a EquidistantCylindrical SRF corresponding to the input parameters.
	Inputs	parameters: <u>EC Parameters</u>
	Outputs	new_srf: EquidistantCylindrical
	Error conditions	1. INVALID_INPUT if parameters are not valid for this SRF.
Method	Name	GetSRFParameters
	Semantics	This method outputs the SRF parameters.
	Inputs	None
	Outputs	parameters: <u>EC Parameters</u>
	Error conditions	No additional error conditions. (See 11.3.x a.)

Standard SRFs

The pre-defined SRFs standardized in clause 8.6 are represented as standard instances of the concrete SRFT class corresponding to the SRFT from which each standard SRF is derived. In particular, each standard SRF object is an instance of an SRFT class. A standard SRF object differs from other instances of its SRFT class in that the GetCodes method of the object outputs the corresponding SRF_Code. The function, CreateStandardSRF3D, specified in Table 0.36 creates and outputs a standard SRF object from an SRF_Code input. An output SRF object corresponds by SRF_Code to Table 8.30.

Table 0.36 — CreateStandardSRF3D

Name	CreateStandardSRF3D
Semantic	The input SRF_Code determines a corresponding SRF label and Table 11.Y entries for SRF class and Create parameters. The function creates and outputs a cooresponding SRF object.
Input	code: <u>SRF_Code</u>
Output	new_srf: <u>BaseSRF3D</u>
Error conditions	3. INVALID_CODE, if the code is not a valid SRF_Code. 4. CREATION_FAILURE, if the Create method of the corresponding SRF class fails.

Example: CreateStandardSRF with input code = 3, produces as output an SRF object corresponding to SRF_GEOCENTRIC_EARTH_1984.

SRF Set Classes

A member of an SRF set for a given ORM is represented as a instance of the SRFT class of the SRF Set. In particular, each SRF Set member object is an instance of an SRFT class. An SRF Set member object differs from other instances of its SRFT class in that the GetCodes method of the object outputs the corresponding SRFS_Code and SRFS_Member_Code. The function CreateSRFSetMember creates and outputs an SRF object in the SRF set corresponding to the input SRFS_Code, ORM_Code and SRFS_Member_Code.

Table 0.37 — CreateSRFSetMember

Name	CreateSRFSetMember
Semantic	The input SRFS_Code, ORM_Code determine a corresponding SRFS and the

	SRFS_Member_Code input determines a member of that SRFS. The function creates and outputs an SRF object corresponding to that SRFS member.
Input	set_code: <u>11.2.5.4 SRFT, SRF, and SRFS and</u> SRFS member codes set_member_code: <u>SRFS Member Code</u> orm_code: <u>ORM Code</u>
Output	new_srf: <u>BaseSRF3D</u>
Error conditions	3. INVALID_CODE, if the set_code is not a valid SRFS_Code, or the orm_code is not a valid ORM_Code for the SRFS, or the set_member_code is not a valid SRFS_Member_Code for the SRFS. 4. CREATION_FAILURE, if the Create method of the corresponding SRF class fails.

Example: CreateStandardSRF with input code = 3, produces as output an SRF object corresponding to SRF_GEOCENTRIC_EARTH_1984.

Object inheritance hierarchy

The object inheritance hierarchy is summarized in Figure 0.1a and b.

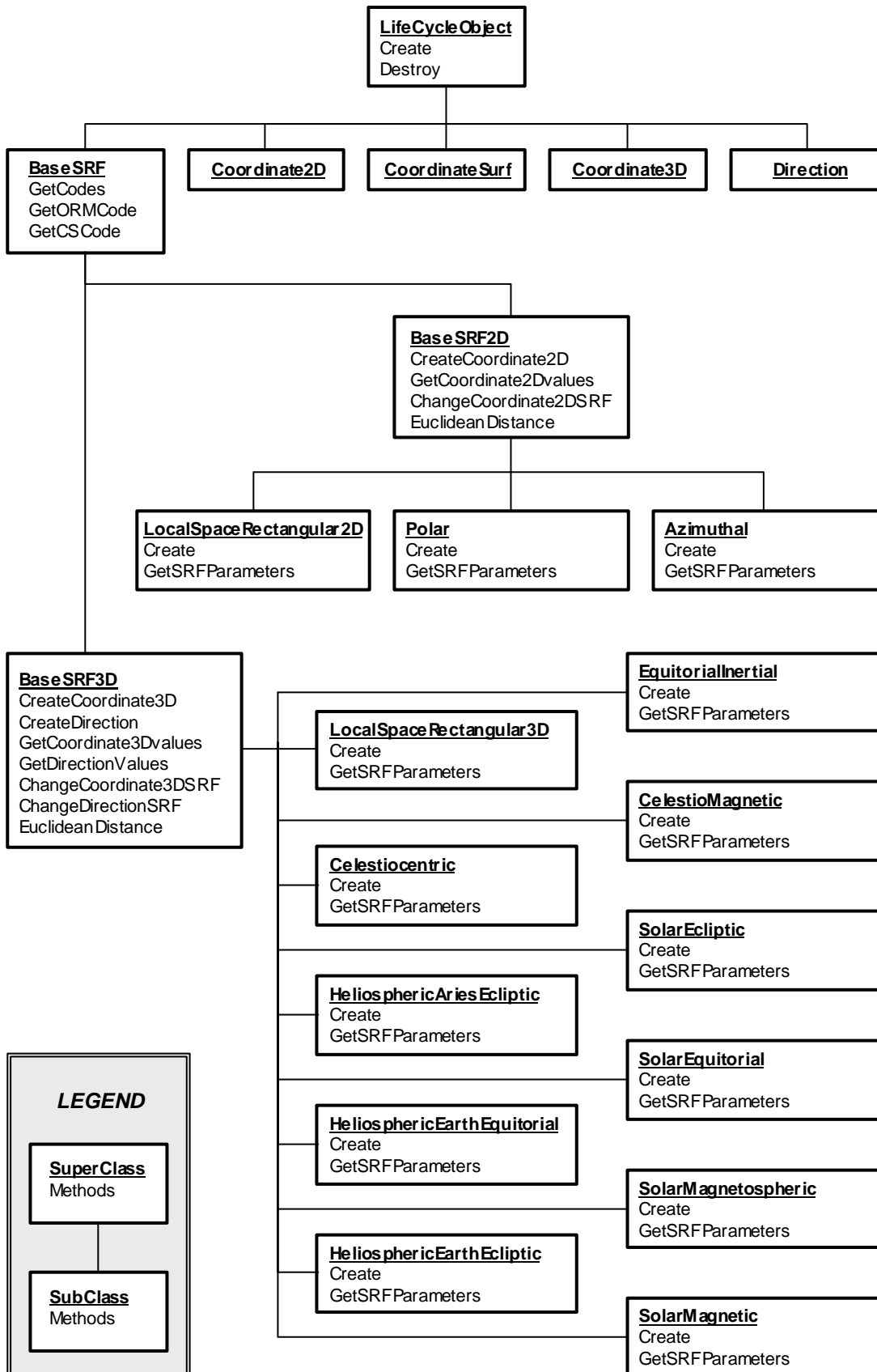


Figure 0.1a — Object inheritance hierarchy

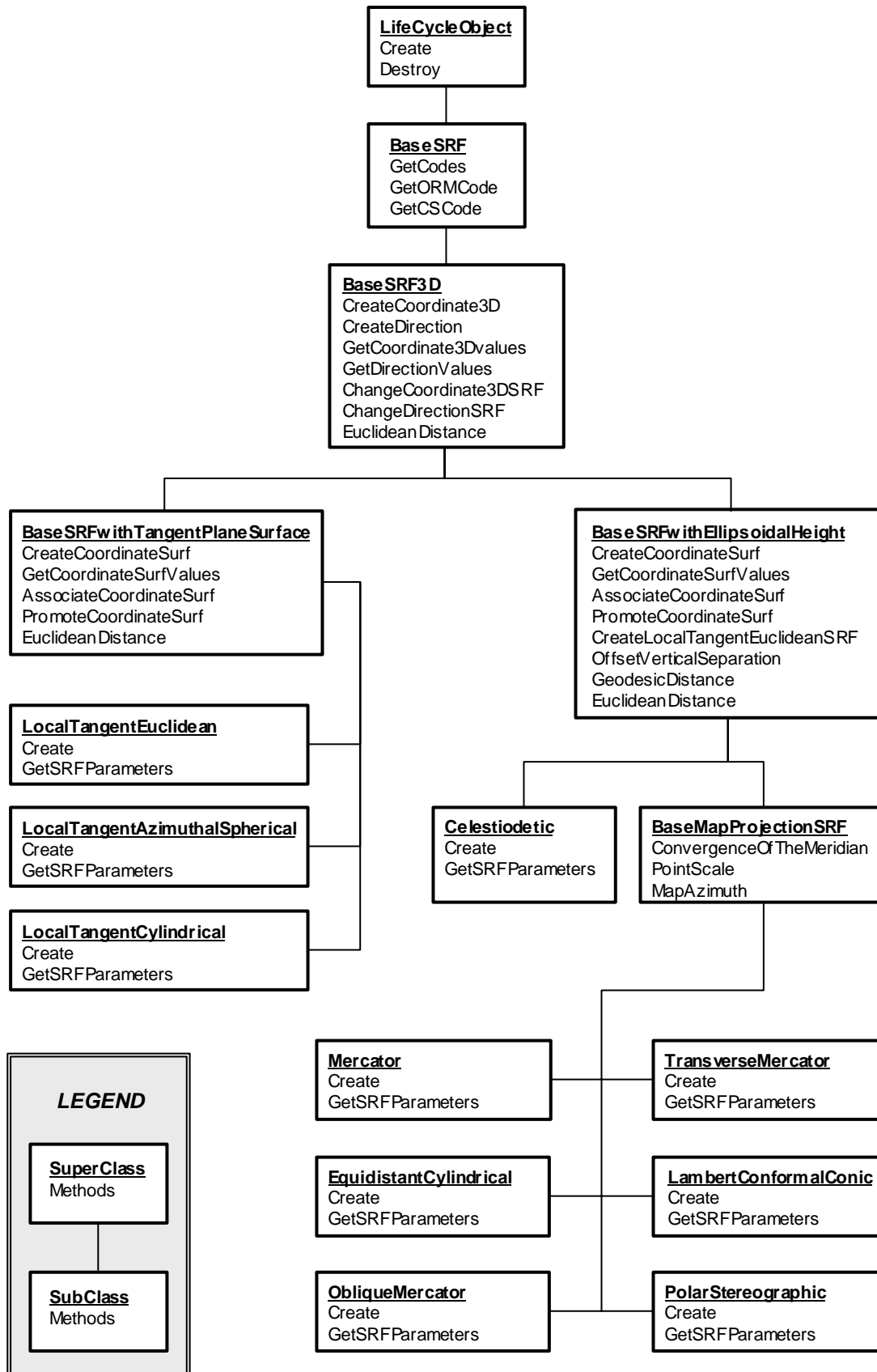


Figure 0.1b — Object inheritance hierarchy

Method precedence for life cycle objects

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

- d. The create method (including subclassed versions) of a life cycle object shall be the first method invoked on any instance of the object.
- e. 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:
 - i. explicit – invoked by the API user,
 - ii. implicit – managed by the runtime system, or
 - iii. implicit/optionally explicit – managed by the runtime system if not explicitly invoked by the API user on any single instance.
- f. All other methods shall only be invoked after the create method and before the destroy method.

Example 1: Find the Euclidean distance between two locations.

```
--Note: Label in italics denotes a symbolic constant for this example --
Celestiodetic method Create( Input: ORM_N_AM_1983_CONUS; Output: srf)
srf method CreateCoordinate3D( Inputs -77°( $\pi/180^\circ$ ), +38°( $\pi/180^\circ$ ), 0; Output: coordinate1)
srf method CreateCoordinate3D( Inputs +3°( $\pi/180^\circ$ ), +49°( $\pi/180^\circ$ ), 0; Output: coordinate2)
srf method EuclideanDistance( Inputs coordinate1, coordinate2, 0; Output: distance)
-- use distance result --
coordinate1 method destroy
coordinate2 method destroy
srf method destroy
```

Example 2: Change SRF representation of a location from UTM to Celestiocentric

```
--Note: Labels in italics denote symbolic constants for this example --
Function CreateSRFSetMember
  ( Input: SRFS_UNIVERSAL_TRANSVERSE_MERCATOR, ORM_N_AM_1983_CONUS,
    ZONE_23_NORTHERN_HEMISPHERE, Output: source_srf)
Function CreateCoordinate3D( Inputs 120, 400, 0, Output: source_coordinate)
Function CreateStandardSRF( Input: SRF_GEOCENTRIC_EARTH_1984, Output: target_srf)
srf method ChangeCoordinate3DSRF( Inputs source_srf, source_coordinate, Output: target_coordinate)
-- use result --
source_coordinate method destroy
target_coordinate method destroy
source_srf method destroy
target_srf method destroy
```