

Open Geospatial Consortium

Submission Date: 2016-04-04

Approval Date: 2016-09-23

Publication Date: 2017-02-23

External identifier of this OGC® document: <http://www.opengis.net/doc/IS/CDB-core/1.0>

Internal reference number of this OGC® document: 15-113r3

Version: 1.0

Category: OGC® Implementation

Editor: Carl Reed, PhD

Volume 1: OGC CDB Core Standard: Model and Physical Data Store Structure

Copyright notice

Copyright © 2017 Open Geospatial Consortium
To obtain additional rights of use, visit <http://www.opengeospatial.org/legal/>.

Warning

This document is an OGC Member approved international standard. This document is available on a royalty free, non-discriminatory basis. Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: OGC® Standard
Document subtype:
Document stage: Approved
Document language: English

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

1.	Introduction.....	17
1.1	Conformance	18
1.2	References	18
1.3	Terms and Definitions.....	18
1.2.1	Abbreviations.....	18
1.4	Conventions.....	19
1.4.1	Identifiers	19
1.4.2	CDB XML Schema Definitions.....	19
1.4.2.1	The CDB Namespace.....	20
1.4.2.2	Schema Conventions.....	20
1.4.3	CDB Metadata Files.....	20
1.4.4	CDB Directory File Naming and Structure	20
1.5	CDB Feature Data Dictionary	21
1.6	Introduction	21
1.6.1	Purpose.....	21
1.6.2	Document Structure	21
1.6.3	What is the CDB Standard: An Overview	22
1.6.4	What the CDB Standard is Not.....	24
1.7	General CDB Data Store and Implementation requirements	25
1.7.1	Platform Requirements	25
1.7.1.1	File System	26
1.7.1.2	Operating System	26
1.7.1.3	Transport Protocols	26
1.7.1.4	System Hardware Independence.....	26
1.7.1.5	Literal Case.....	26
1.7.2	General Data Representation Requirements	27
1.7.2.1	Compression of Storage Intensive Imagery Datasets	27
1.7.2.2	Compression of other imagery datasets.....	28
1.7.2.3	Units of Measure	28
1.7.3	Coordinate Reference Systems	28
1.7.4	Geographic Coordinates.....	29

2.	CDB Concepts	29
2.1	Characteristics of the CDB tiling storage model	30
	Geocell extent.....	31
	Geocell length.....	31
2.1.1	Details of the Tiling System in the CDB core model	31
2.1.2	Tile Levels of Detail (Tile LoD).....	34
2.1.2.1	Tile LoD Area Coverage Rules.....	39
2.1.2.2	Tile-LOD Hierarchy Rules.....	40
2.1.2.3	Tile-LOD Replacement Rules.....	41
2.1.3	Handling of the North and South Pole.....	41
2.2	File System Requirements.....	42
2.3	Light Naming	42
2.3.1	Adding Names to the CDB Light Name Hierarchy	44
2.4	Model Component Naming.....	45
2.4.1	Adding New Model Components	46
2.5	Materials.....	46
2.5.1	Base Materials.....	48
2.5.2	Composite Materials.....	49
2.5.2.1	Composite Material Substrates	49
2.5.2.2	Composite Material Tables (CMT)	51
2.5.2.3	Example 1.....	53
2.5.2.4	Example 2.....	53
2.5.3	Bringing it all Together.....	53
2.5.4	Determination of Material Properties by Sensor Environmental Model (SEM)	54
2.5.5	Generation of Materials for Inclusion in CDB Datasets.....	55
3.	CDB Structure.....	56
3.3	CDB Model Types	66
3.3.1	GTModel (Geotypical 3D Model)	67
3.3.2	GSModel (Geospecific 3D Model).....	67
3.3.3	T2DModel (Tiled 2D Model)	68
3.3.4	MModel (Moving 3D Model).....	68
3.3.5	Use of GSModels and GTModels.....	68

3.3.6	Organizing Models into Levels of Detail.....	70
3.3.7	Organizing Models into Datasets.....	73
3.3.8	Terms and Expressions	74
3.3.8.1	Feature Classification.....	74
3.3.8.1.1	A note on Feature Codes	74
3.3.8.2	Model Name	75
3.3.8.3	DIS Entity Type.....	75
3.3.8.4	Texture Name	76
3.3.8.5	Level of Detail	77
3.4	GTModel Library Datasets.....	77
3.4.1	GTModel Directory Structure 1: Geometry and Descriptor.....	78
3.4.1.1	GTModelGeometry Entry File Naming Convention	79
3.4.1.2	GTModelGeometry Level of Detail Naming Convention.....	80
3.4.1.3	GTModelDescriptor Naming Convention.....	80
3.4.1.4	Examples.....	81
3.4.2	GTModel Directory Structure 2: Texture, Material, and CMT	82
3.4.2.1	GTModelTexture Naming Convention	83
3.4.2.2	GTModelMaterial Naming Convention.....	83
3.4.2.3	GTModelCMT Naming Convention	84
3.4.2.4	Examples.....	85
3.4.3	GTModel Directory Structure 3: Interior Geometry and Descriptor.....	86
3.4.3.1	GTModelInteriorGeometry Naming Convention	87
3.4.3.2	GTModelInteriorDescriptor Naming Convention.....	88
3.4.3.3	Examples.....	88
3.4.4	GTModel Directory Structure 4: Interior Texture, Material, and CMT	89
3.4.4.1	GTModelInteriorTexture Naming Convention	90
3.4.4.2	GTModelInteriorMaterial Naming Convention.....	91
3.4.4.3	Example 1.....	91
3.4.4.4	Example 2.....	92
3.4.5	GTModel Directory Structure 5: Signature	93
3.4.5.1	GTModelSignature Naming Convention	94

3.4.5.2	Examples	95
3.4.6	GTModel Complete Examples	95
3.5	MModel Library Datasets	96
3.5.1	MModel Directory Structure 1: Geometry and Descriptor	97
3.5.1.1	MModelGeometry Naming Convention	98
3.5.1.2	MModelDescriptor Naming Convention	98
3.5.1.3	Examples	99
3.5.2	MModel Directory Structure 2: Texture, Material, and CMT	99
3.5.2.1	MModelTexture Naming Convention	100
3.5.2.2	MModelMaterial Naming Convention	101
3.5.2.3	MModelCMT Naming Convention	101
3.5.2.4	Examples	102
3.5.3	MModel Directory Structure 3: Signature	102
3.5.3.1	Naming Convention	103
3.5.3.2	Examples	104
3.5.4	MModel Complete Examples	105
3.6	CDB Tiled Datasets	105
3.6.1	Tiled Dataset Types	106
3.6.1.1	Raster Datasets	106
3.6.1.2	Vector Datasets	106
3.6.1.3	Model Datasets	108
3.6.2	Tiled Dataset Directory Structure	108
3.6.2.1	Directory Level 1 (Latitude Directory)	110
3.6.2.2	Directory Level 2 (Longitude Directory)	111
3.6.2.3	Directory Level 3 (Dataset Directory)	114
3.6.2.4	Directory Level 4 (LOD Directory)	115
3.6.2.5	Directory Level 5 (UREF Directory)	115
3.6.3	Tiled Dataset File Naming Conventions	117
3.6.3.1	File Naming Convention for Files in Leaf Directories (UREF Directory)	117
3.6.3.2	File Naming Convention for Files in ZIP Archives	120
3.7	Navigation Library Dataset	123

3.7.1	Navdata Structure.....	123
3.7.2	Navigation Data Naming Convention.....	124
3.7.2.1	Examples.....	124
4.	CDB File Formats.....	124
5.	CDB Datasets.....	127
5.1	Metadata Datasets	127
5.1.1	Light Data Hierarchy Metadata	128
5.1.1.1	Client Specific Lights Definition Metadata	131
5.1.2	Model Components Definition File	134
5.1.3	Base Materials Table.....	134
5.1.4	Default Values Definition Table.....	135
5.1.5	No content.....	136
5.1.6	Version Metadata	136
5.1.7	CDB Attributes Metadata	137
5.1.7.1	Definition of the <Attribute> Element	138
5.1.7.2	Definition of the <Unit> Element.....	139
5.1.7.3	Definition of the <Scaler> Element	140
5.1.7.4	Example of CDB_Attributes.xml.....	141
5.1.8	Geomatics Attributes Metadata.....	142
5.1.9	Vendor Attributes Metadata.....	142
5.1.10	Configuration Metadata	143
5.1.10.1	A Note about Folder Path	143
5.1.10.2	Example	143
5.2	Navigation Library Datasets.....	144
5.2.1	Schema Files	146
5.2.1.1	Example	147
5.2.2	Key Datasets	149
5.2.2.1	Example	149
5.3	CDB Model Textures	151
5.4	GTModel Library Datasets.....	153
5.5	MModel Library Datasets	155
5.6	Tiled Raster Datasets.....	157

5.6.1	Tiled Elevation Dataset.....	163
5.6.1.1	Terrain Mesh Types	165
5.6.1.2	List of all Elevation Dataset Components	165
5.6.1.3	Primary Terrain Elevation Component	167
5.6.1.4	Primary Alternate Terrain Elevation Component	169
5.6.1.5	Terrain Constraints	172
5.6.1.6	MinElevation and MaxElevation Components.....	177
5.6.1.7	MaxCulture Component	186
5.6.1.8	Subordinate Bathymetry Component.....	188
5.6.1.9	Subordinate Alternate Bathymetry Component.....	192
5.6.1.10	Subordinate Tide Component.....	193
5.6.2	Tiled Imagery Dataset.....	197
5.6.2.1	Raster-Based Imagery File Storage Extension Naming	197
5.6.2.2	List of all Imagery Dataset Components	204
5.6.2.3	Visible Spectrum Terrain Imagery (VSTI) Components	205
5.6.2.4	Visible Spectrum Terrain Light Map (VSTLM) Component	209
5.6.3	Tiled Raster Material Dataset	211
5.6.3.1	List of all Raster Material Dataset Components	213
5.6.3.2	Composite Material Index Component.....	214
5.6.3.3	Composite Material Mixture Component.....	216
5.6.3.4	Composite Material Table Component.....	216
5.7	Tiled Vector Datasets.....	217
5.7.1	Introduction to Vector Datasets	218
5.7.1.1	Vector Type Usage and Conventions	221
5.7.1.2	CDB Attribution.....	223
5.7.1.3	CDB Attributes	232
5.7.1.4	Explicitly Modeled Representations	264
5.7.1.5	Implicitly Modeled Representations.....	265
5.7.1.6	Handling of Topological Networks.....	265
5.7.1.7	Handling of Light Points.....	269
5.7.1.8	Allocation of CDB Attributes To Vector Datasets.....	269

5.7.2	Tiled Navigation Dataset	272
5.7.2.1	Default Read Value	275
5.7.2.2	Default Write Value	275
5.7.3	Tiled GSFeature Dataset	275
5.7.3.1	Default Read Value	276
5.7.3.2	Default Write Value	276
5.7.4	Tiled GTFeature Dataset	277
5.7.4.1	Default Read Value	278
5.7.4.2	Default Write Value	278
5.7.5	Tiled GeoPolitical Feature Dataset	278
5.7.5.1	Boundary and Location Features	279
5.7.5.2	Elevation Constraint Features	280
5.7.5.3	Default Read Value	282
5.7.5.4	Default Write Value	282
5.7.6	Tiled RoadNetwork Dataset	282
5.7.6.1	Default Read Value	284
5.7.6.2	Default Write Value	284
5.7.7	Tiled RailRoadNetwork Dataset	284
5.7.7.1	Default Read Value	285
5.7.7.2	Default Write Value	286
5.7.8	Tiled PowerLineNetwork Dataset	286
5.7.8.1	Default Read Value	287
5.7.8.2	Default Write Value	287
5.7.9	Tiled HydrographyNetwork Dataset	287
5.7.9.1	Default Read Value	289
5.7.9.2	Default Write Value	289
5.7.10	Tiled Vector Composite Material Table (VCMT)	289
5.7.10.1	Data Type	290
5.7.10.2	Default Read Value	290
5.7.10.3	Default Write Value	290
5.8	Tiled Model Datasets	290

5.8.1	Tiled GSModel Datasets	290
5.8.1.1	GSModel Archive Size Limit	292
5.8.2	Tiled T2DModel Datasets	292
Abstract Conformance Test Suite		293
A.1	Conformance Test Class: OGC CDB Core Standard (Core)	293
A.1.1	General CDB Data Store and Implementation	293
A.1.2	Platform	293
A.1.3	General Data Representation	294
A.1.4	Structure-Tiling Model	295
A.1.5	Light Naming	297
A.1.6	Light Name Hierarchy	298
A.1.7	Materials	299
A.1.8	CDB Root Directory	301
A.1.9	Version Metadata File	302
A.1.10	FLIR Metadata File	302
A.1.11	Data Store Version Directory Structure	303
A.1.12	Model Types	304
A.1.13	Geospecific Model (GSModel) Storage	304
A.1.14	Geotypical Models (GTModel) Naming Conventions	305
A.1.15	Moving Model (MModel) Naming Conventions	309
A.1.16	Tiled Datasets	311
A.1.17	Archive Names	312
A.1.18	NavData Naming Convention	314
A.1.19	Metadata Datasets	314
A.1.20	Navigation Data	316
A.1.21	Tiled Raster Datasets	317
A.1.22	Tiled Elevation Dataset	318
A.1.23	Tiled Terrain Bathymetry	321
A.1.24	Tiled JPEG Metadata	322
A.1.25	Visible Spectrum Terrain Imagery (VSTI)	322
A.1.26	Visible Spectrum Terrain Light Map (VSTLM)	323
A.1.27	Raster Composite Material	323
A.1.28	Tiled Vector Datasets	324

A.1.29 Vector Datasets Mandatory Attribute Usage	326
A.1.30 Vector Datasets Topology	327
A.1.31 Elevation Constraints	328
A.1.32 Tiled Road Networks	329
A.1.33 Tiled Railroad Networks	329
A.1.34 Tiled PowerLine Networks	330
A.1.35 Tiled Hydrography Networks	330
A.1.36 Vector Composite Material Table (VCMT)	331

Figures

Figure 2- 1: CDB Earth Slice Representation.....	32
Figure 2- 2: Variation of Longitudinal Dimensions version Latitude	33
Figure 2- 3: Tile-LOD Hierarchy.....	38
Figure 2- 4: Earth Slice Example (Five Levels of Detail)	40
Figure 2- 5: Extract from Light Naming Hierarchy.....	43
Figure 2- 6: Seabed Composite Material	50
Figure 2- 7: Oil Tank Composite Materials.....	50
Figure 2- 8: Thermal Simulation of Oil Tank Composite Materials	51
Figure 2- 9: Flow of Material Attribution Data	54
Figure 3- 1: UML Diagram of CDB Version Concept	60
Figure 3- 2: A Valid Chain of CDB Versions.....	60
Figure 3- 3: UML Diagram of CDB Extension Concept.....	61
Figure 3- 4: Adding Content to the CDB data store	63
Figure 3- 5: Modifying Content of the CDB Data Store	64
Figure 3- 6: Deleting Content from a CDB Data Store	64
Figure 3- 7: UML Diagram of CDB Configuration Concept	65
Figure 3- 8: Allocation of CDB Geocells with Increasing Latitude	113
Figure 5- 1: Center Grid Data Elements	160
Figure 5- 2: Corner Grid Elements	161
Figure 5- 3: Center Data Elements as a Function of LODs	161
Figure 5- 4: Corner Data Elements as a Function of LODs.....	162
Figure 5- 5: Example of Digital Elevation Model (DEM).....	163
Figure 5- 6: DEM Depicted as a Grid of Elevations at Regular Sample Points	164
Figure 5-7: CDB Mesh Types	
Figure 5-8: Primary Terrain Elevation Component Figure 5-9: Modeling of Wells, Overhanging Cliffs and Tunnels	

Figure 5-10: Encoding of Lat/Long Offsets
Figure 5-11: Grid Element Coverage within a CDB Tile
Figure 5-12: Storage Tank Point-Feature
Figure 5-13: Road Lineal Feature)
Figure 5-14: LOD Structure of Raster Datasets
Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D)
Figure 5-16: Generation of LODs for the MinMaxElevation Dataset (2D)
Figure 5-17: Generation of LODs for the MinMaxElevation Dataset (1D) – Special Case
Figure 5-18: Availability of LODs for Elevation and MinMaxElevation Datasets
Figure 5-19: Missing MinMaxElevation Datasets
Figure 5-20: Primary Terrain Elevation and Subordinate Bathymetry Components
Figure 5-21: Derived Earth Elevation Values (OK)
Figure 5-22: Example of Primary Terrain Elevation and Bathymetry Components
Figure 5-23: Terrain Elevation, Bathymetry and Tide Components
Figure 5-24: Derived Earth Elevation, Water Elevation and Surface Elevation Values
Figure 5-25: Projection of Terrain Imagery Dataset onto Terrain Elevation Dataset
Figure 5-26: Image Classification Example
Figure 5-27: Example of a Raster Material Dataset
Figure 5-28: Instance-level Attribution Schema
Figure 5-29: Class-level Attribution Schema
Figure 5-30: Relation between Shapes and Attributes
Figure 5-31: RTAI Typical Usage Histogram
Figure 5-32: Example of a Topological Network
Figure 5-33: Example of International Boundaries
Figure 5-34: Example of City Locations

i. **Abstract**

The CDB standard defines a standardized model and structure for a single, versionable, virtual representation of the earth. A CDB structured data store provides for a geospatial content and model definition repository that is plug-and-play interoperable between database authoring workstations. Moreover, a CDB structured data store can be used as a common online (or runtime) repository from which various simulator client-devices can simultaneously retrieve and modify, in real-time, relevant information to perform their respective runtime simulation tasks. In this case, a CDB is plug-and-play interoperable between CDB-compliant simulators. A CDB can be readily used by existing simulation client-devices (legacy Image Generators, Radar simulator, Computer Generated Forces, etc.) through a data publishing process that is performed on-demand in real-time.

The application of CDB to future simulation architectures will significantly reduce runtime-source level and algorithmic correlation errors, while reducing development, update and configuration management timelines. With the addition of the High Level Architecture - - Federation Object Model (HLA/FOM)¹ and DIS protocols, the application of the CDB standard provides a Common Environment to which inter-connected simulators share a common view of the simulated environment.

The CDB standard defines an open format for the storage, access and modification of a synthetic environment database. A **synthetic environment** is a [computer simulation](#) that represents activities at a high level of realism, from simulation of theaters of war to factories and manufacturing processes. These environments may be created within a single computer or a vast distributed network connected by local and wide area networks and augmented by super-realistic special effects and accurate behavioral models. SE allows visualization of and immersion into the environment being simulated².

This standard defines the organization and storage structure of a worldwide synthetic representation of the earth as well as the conventions necessary to support all of the subsystems of a full-mission simulator. The standard makes use of several commercial and simulation data formats endorsed by leaders of the database tools industry. A series of associated OGC Best Practice documents define rules and guidelines for data representation of real world features.

The CDB synthetic environment is a representation of the natural environment including external features such as man-made structures and systems. A CDB data store can include terrain relief, terrain imagery, three-dimensional (3D) models of natural and man-made cultural features, 3D models of dynamic vehicles, the ocean surface, and the ocean bottom, including features (both

¹ https://en.wikipedia.org/wiki/High-level_architecture

² "Department of Defense Modeling and Simulation (M&S) Glossary", DoD 5000.59-M,

natural and man-made) on the ocean floor. In addition, the data store can include the specific attributes of the synthetic environment data as well as their relationships.

The associated CDB Standard Best Practice documents provide a description of a data schema for Synthetic Environmental information (i.e. it merely describes data) for use in simulation. The CDB Standard provides a rigorous definition of the semantic meaning for each dataset, each attribute and establishes the structure/organization of that data as a schema comprised of a folder hierarchy and files with internal (industry-standard) formats.

A CDB conformant data store contains datasets organized in layers, tiles and levels-of-detail. Together, these datasets represent the features of a synthetic environment for the purposes of distributed simulation applications. The organization of the synthetic environmental data in a CDB compliant data store is specifically tailored for real-time applications.

ii. **Keywords**

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, CDB, Common Data Base, simulation, synthetic environment, virtual

iii. **Preface**

The industry-maintained CDB model and data store structure has been discussed and demonstrated at OGC Technical Committee meetings beginning in September, 2013.

At the suggestion of several attendees at the first OGC CDB ad-hoc meeting in September, 2014, the existing CDB standard was slightly reformatted and approved as an OGC Best Practice in May, 2015.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

iv. **Submitting organizations**

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

CAE Inc.
Carl Reed, OGC Individual Member
Envitia, Ltd

Glen Johnson, OGC Individual Member
KaDSci, LLC
Laval University
Open Site Plan
University of Calgary
UK Met Office

The OGC CDB standard is based on and derived from an industry developed and maintained specification, which has been approved and published as OGC Document 15-003: OGC Common Data Base Volume 1 Main Body. An extensive listing of contributors to the legacy industry-led CDB specification is contained in Chapter 11, pp 475-476 in that OGC Best Practices Document (https://portal.opengeospatial.org/files/?artifact_id=61935).

v. **Submitters**

All questions regarding this submission should be directed to the editor or the submitters:

Name	Affiliation
Carl Reed	Carl Reed & Associates (Editor)
David Graham	CAE Inc.

vi. **Future Work**

The CDB community anticipates that additional standardization work will be required to prescribe content appropriate to targeted simulation applications, new use cases, and application in new domains. In its current form, the CDB standard does not mandate synthetic environmental richness, quality and resolution.

Further, the OGC CDB Standards Working Group (SWG) members understand there is a requirement for eventual alignment of the CDB standard with the OGC/ISO standards baseline. In Version 1 of the CDB standard, effort was invested to begin aligning terminology and concepts, specifically in the coordinate reference system discussions and requirements.

The current version of the CDB standard is fully backwards compatible with version 3.2 of the CDB specification as defined and implemented by the current CDB implementer and user community. The requirements for a CDB data store are focused on the ability to store, manage, and access extremely large volumes of geographic content and related model definitions. In this version of the standard, initial harmonization with the OGC and ISO standards baseline has begun. For example, where appropriate, the CDB simulation community terms and definitions have been replaced with OGC/ISO terms and definitions. Further, the standards documents have

been reorganized and structured to be consistent with the OGC Modular Specification Policy. However, the CDB SWG and community recognize the need to further harmonize and align this standard with the OGC/ISO baseline and other IT best practices. There has already been considerable discussion in this regard.

Based on such discussions and comments received during the public comment period, the following future work tasks are envisioned:

1. Describe explicitly how the CDB model may or may not align with the OGC DGGS standard;
2. Provide best practice details on how to use WMS, WFS, and WCS to access existing CDB data stores. This work may require Interoperability Experiments to better understand the implications of these decisions;
3. Extend the supported encodings and formats for a CDB structured data store to include the use of the OGC GeoPackage, CityGML, and InDoorGML standards as well as other broadly used community encoding standards, such as GeoTIFF. This work may require performing OGC Interoperability Experiments to better understand the implications of these decisions.
4. Further align CDB terminology to be fully consistent with OGC/ISO terminology.

Making these enhancements will allow the use and implementation of a CDB structured data store for application areas other than aviation simulators.

vii. A note on using a CDB Data Store with OGC Standards

Please refer to Volume 0: CDB Primer, Clause 5 for an operational example of using OGC standards to query, access, and modify content in a CDB data store.

1. Introduction

The CDB standard defines a data model and the representation, organization, storage structure and conventions necessary to support all of the subsystems of a simulation workflow. The standard makes use of existing commercial and simulation data formats³.

The CDB conceptual model is a representation of the natural environment including external features such as man-made structures and systems. The model encompasses planimetry, terrain relief, terrain imagery, three-dimensional (3D) models of natural and man-made cultural features, 3D models of vehicles, the ocean surface, and the ocean bottom, including features (both natural and man-made) on the ocean floor. In addition, the model includes the attributes of the synthetic environment data as well as their relationships.

A data store that conforms to the CDB standard (i.e., a CDB) contains datasets organized in layers, tiles and levels-of-detail. Together, these datasets represent the features and models of a synthetic environment for the purposes of distributed simulation applications. The organization of the geospatial data in a CDB data store is specifically tailored for real-time applications.

For ease of editing and review, the standard has been separated into 12 Volumes and a schema repository.

- Volume 0: OGC CDB Companion Primer for the CDB standard. (Best Practice)
- Volume 1: OGC CDB Core Standard: Model and Physical Data Store Structure. The main body (core) of the CDB standard (Normative).
- Volume 2: OGC CDB Core Model and Physical Structure Annexes (Best Practice).
- Volume 3: OGC CDB Terms and Definitions (Normative).
- Volume 4: OGC CDB Best Practice use of Shapefiles for Vector Data Storage (Best Practice).
- Volume 5: OGC CDB Radar Cross Section (RCS) Models (Best Practice Best Practice).
- Volume 6: OGC CDB Rules for Encoding Data using OpenFlight (Best Practice).
- Volume 7: OGC CDB Data Model Guidance (Best Practice).
- Volume 8: OGC CDB Spatial Reference System Guidance (Best Practice).
- Volume 9: OGC CDB Schema Package: provides the normative schemas for key features types required in the synthetic modelling environment. Essentially, these schemas are designed to enable semantic interoperability within the simulation context. (Normative)
- Volume 10: OGC CDB Implementation Guidance (Best Practice).
- Volume 11: OGC CDB Core Standard Conceptual Model (Normative)
- Volume 12: OGC CDB Navais Attribution and Navais Attribution Enumeration Values (Best Practice)

³ Future versions of the CDB standard will define how to use other industry approved standards and formats.

Once this version is approved as an OGC standard, the CDB SWG shall consider how this standard will evolve to align with the OGC/ISO standards baseline. Discussion and suggestions have already started, specifically in relation to the OGC GeoPackage, DGGS, KML, and CityGML standards.

1.1 Conformance

This standard defines an Abstract Test Suite in Annex A.

Requirements for one standardization target type are considered.

Conformance with this standard shall be checked using all the relevant tests specified in Annex A (normative) of this document. The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in the OGC Compliance Testing Policies and Procedures and the OGC Compliance Testing web site⁴.

1.2 References

Please see the Annex B for a complete list of all normative and informative references.

1.3 Terms and Definitions

This document uses the terms defined in Sub-clause 5.3 of [OGC 06-121r8], which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this standard.

Terms and Definitions specific to this standard or defined by existing ISO 19000 series standards are provided in Volume 3: OGC CDB Terms and Definitions <need URL at publication>.

1.2.1 Abbreviations

AO1 Angle of Orientation with greater than 1 degree resolution. The angular distance measured from true north (0 deg) clockwise to the major (Y) axis of the feature.

DGIWG Defence Geographic Information Working Group

DIS Distributed Interactive Simulations [IEEE Std 1278TM]

FC Feature Code

FDD Feature Data Dictionary

FOM Federation Object Model

HLA High Level Architecture [IEEE Std 1516TM]

⁴ www.opengeospatial.org/cite

1.4 Conventions

This sections provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

1.4.1 Identifiers

The normative provisions in this standard are denoted by the URI

`http://www.opengis.net/spec/cdb/1.0/core/{requirement}`

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

For the sake of brevity, the use of “req” in a requirement URI denotes:

<http://www.opengis.net/spec/cdb/1.0>

An example might be:

`req/core/file-system`

1.4.2 CDB XML Schema Definitions.

NOTE: The content in this clause was originally in CDB Volume 2, Annex J.

The CDB standard makes an extensive use of XML to describe several parts of the standard. XML is used to describe CDB metadata, to store global datasets, to add attributes and information to 3d models, such as an OpenFlight model, to describe base and composite materials, etc.

The following XML Schemas can be found in the `\CDB\Metadata\Schema` subdirectory of the CDB Standard Distribution Package:

- `Base_Material_Table.xsd`
- `Composite_Material_Table.xsd`
- `Configuration.xsd`
- `Defaults.xsd`
- `Feature_Data_Dictionary.xsd`
- `Lights.xsd`
- `Lights_Tuning.xsd`
- `Model_Components.xsd`
- `Model_Metadata.xsd`
- `OpenFlight_Model_Extensions.xsd`
- `Vector_Attributes.xsd`

- Version.xsd

1.4.2.1 The CDB Namespace

The CDB standard makes use of several XML namespaces to isolate the definitions of its schemas. The name of these namespaces is built around a base URL.

1.4.2.2 Schema Conventions

The target namespace of all CDB schemas follow this pattern:

```
"http://www.opengis.net/cdb/[Name]/[Version]"
```

Where the *Name* is identical to the filename portion of the file containing the schema and *Version* is the version number of the schema.

To illustrate how a target namespace is composed, here is the target namespace of the schema found in Version.xsd (item 12 in the list above):

```
"http://www.opengis.net/cdb/Version/1.0"
```

IMPORTANT NOTE: For brevity, the literal “CDB” in a schema path should be expanded to:

```
http:// schemas.opengis.net/cdb/[Name]/1.0
```

in any implementation.

1.4.3 CDB Metadata Files

There are a number of CDB XML files that are referenced from the CDB metadata folder.

- \CDB\Metadata\Lights.xml - CDB Lights
- \CDB\Metadata\Defaults.xml - Default Read and Write values to be used by Simulator Client-Devices

1.4.4 CDB Directory File Naming and Structure

The CDB directory and folder structure is defined by a combination of folder hierarchy and metadata files delivered with the CDB Standard Distribution Package.

The /CDB folder hierarchy provides a complete list of directory and filename patterns of the CDB; it summarizes the structure of the CDB presented in chapter 3 of this document. The following files are necessary to expand the patterns:

- /CDB/Metadata/Feature_Data_Dictionary.xml provides the list of directory names associated with feature codes.
- /CDB/Metadata/Moving_Model_Codes.xml provides the list of names for DIS Entity Kinds, Domains, and Categories.
- /CDB/Metadata/DIS_Country_Codes.xml contains the list of DIS Country Names.

Together, these files provide all the information required to build the names of all directories permitted by the standard.

1.5 CDB Feature Data Dictionary

The CDB Feature Data Dictionary (FDD) is provided with the CDB standard in the form of an XML file. An XML Stylesheet is provided to format and display the dictionary inside a standard Web browser. Furthermore, the XML Schema defining the format of the FDD can also be found in the Schema subdirectory of the CDB Schema Distribution Package.

See /CDB/Metadata/Feature_Data_Dictionary.xml for the complete list of the supported codes⁵.

Please see section [3.3.8.1](#) for more detailed information on the use of feature codes and extensions to that codelist in the CDB standard.

1.6 Introduction

1.6.1 Purpose

This standard provides a full description of a data model (aka schema) for the synthetic representation of the world. The representation of the synthetic environment in the CDB model as expressed in a physical data store is intended for use by authoring tools and by various simulator client-devices that are able to simultaneously retrieve, in real-time, relevant information to perform their respective runtime simulation tasks. With the addition of the DIS protocol, the application of the CDB standard provides a Common Environment to which interconnected simulators share a common view of the simulated environment.

1.6.2 Document Structure

This document is structured as follows.

Section 1.6 defines general CDB data store and implementation requirements

Sections 2.1, 2.2, 2.3, and 2.4 provide an overview of key elements of the CDB data store structure.

Section 2.5: CDB concepts and semantics. Describes the naming and handling of materials that make up the synthetic environment

Section 3.0: Focuses on aspects of the data model that relate to the structure of the data store repository on the storage subsystem. The organization of the CDB data into tiles, levels-of-detail and datasets is embodied through a set of conventions that prescribe the CDB directory hierarchy and file naming conventions.

Section 4: CDB File Formats provides a description of all the formats prescribed by the CDB Specification.

⁵ Currently, these are FACC codes, Future revisions of the CDB standard shall provide guidance on using other feature code lists.

Section 5: CDB Datasets provides a detailed description of all CDB datasets.

The current CDB standard relies on established industry formats:

- The TIFF format. TIFF encoding rules are defined in Volume 10: OGC CDB Implementation Guidance;
- The Best Practice use of the OpenFlight⁶ format (Volume 6: OGC CDB Rules for Encoding Data using OpenFlight);
- The RGB format. Included in the Gamma Tutorial Section, Volume 10: OGC CDB Implementation Guidance
- The Best Practice use of the Shapefile format in a CDB data store. The Shapefile table content encoding rules are in Volume 4. Volume 4: OGC CDB Best Practice use of Shapefiles for Vector Data Storage;
- JPEG 2000 file format (Volume 2: OGC CDB Core Model and Physical Structure Annexes, Annex H).

Each of these documents has been annotated to reflect the conventions established by the CDB standard. The Best Practice conventions currently define how TIFF, OpenFlight, RGB, Shapefile and JPEG 2000 formatted files are to be interpreted by CDB-compliant simulator readers. Additional encoding formats and conventions will be defined for future versions of the CDB standard.

Annexes J and F of Volume 2 provide the CDB light type naming hierarchy and the CDB model component hierarchies respectively while \CDB\Metadata\Materials.xml provides the material list for the CDB standard.

Other Annexes in Volume 2 further describe additional aspects of the CDB standard:

- Providing the CDB Directory Naming and Structure (Annex M),
- List of Texture Component Selectors (Annex O see footnote 32⁷),
- SGI Image File Format (<http://paulbourke.net/dataformats/sgirgb/sgiversion.html>),
- Table of Dataset Codes (Annex Q⁸)
- How some datasets are derived from others (Annex R see footnote 32).

1.6.3 What is the CDB Standard: An Overview

The CDB standard defines a conceptual model that models the organization, and storage structure of a data store to support real time simulation applications. A data store that conforms to the CDB standard contains datasets organized in layers and tiles that represent the features of the earth for the purposes of distributed simulation applications. A CDB data store can be readily used by existing simulation client-devices (legacy IGs, Radars, CGF, etc.) through a

⁶ Other 3d formats will be evaluated and considered for best practice specification in future versions of CDB. These include CityGML, InDoorGML, COLLADA, and so forth.

⁷ Annex O can be found in Volume 2 CDB Core Model and Physical Structure Annexes

⁸ OGC CDB Core: Model and Physical Structure: Informative Annexes

publishing process performed in real-time. The CDB conceptual model would allow an implementation if a CDB compliant data store in a relational database. However, the data structures used in CDB structured data stores are somewhat different than those used in relational databases because 1.) of the use of standardized data formats adopted by the simulation community and 2.) the CDB storage structure is optimized for near real time simulation applications. The approach defined in this CDB Core standard facilitates the work required to adapt existing authoring tools to read/write/modify data into the CDB and the task to develop runtime publishers (RTP) designed to operate on these data structures.

The CDB standard is fundamentally about:

- A representation of the earth and man-made environment for use in real time simulations.
- A turnkey, as-is representation of the Synthetic Environment (SE) for use in real-time distributed simulation applications.

Currently, the majority of a CDB conformant internal data storage representation is based on five well known and supported data formats endorsed by leaders of the simulation database tools industry. The CDB Best Practices associated with the Core standard are currently recommended for implementation of a CDB data store:

- For the representation of terrain altimetry, terrain surface characteristics relevant to simulation: TIFF/GeoTIFF
- For the representation of 3D culture and moving models: OpenFlight
- For the textures associated with 3D culture and moving models RGB
- For the instancing and attribution of statically positioned point, lineal and polygon 2D/3D culture features.
- For a representation of terrain raster imagery comprising a well-defined and accepted compression method that allows both lossy and lossless schemes: JPEG 2000.

Please note that the OGC CDB Standards Working Group is considering developing best practice guidance for using other industry standard formats and encodings, such as OGC CityGML, OGC InDoorGML, and OGC GeoPackage.

NOTE: Due to the real time requirement, the CDB standard limits the number of units of measure for each physical quantity. For instance, all coordinates are represented in latitude longitude and all distances are in meters.

NOTE: In the future, the CDB standard will evolve to enable the use of other international and de-facto geospatial encoding structures.

The CDB specified storage structure allows efficient searching, retrieval and storage of any information contained within a CDB structured data store. Storage structure aspects include descriptions of each information field used within CDB conformant files, including data types and data type descriptions.

The CDB standard relies on three important concepts to organize the geospatial data:

- **Tiles:** The CDB defined storage structure allows efficient searching, retrieval and storage of any information contained within the CDB. The storage structure portion of this standard geographically divides the world into geodetic tiles (each tile bounded by latitude and longitude), each tile containing a specific set of features (such as terrain altimetry, vectors) and models (such as 3d models and radar cross section models), which are in turn represented by the datasets. The datasets define the basic storage unit used in a CDB data store. The geographic granularity is at the tile level while the information granularity is at the dataset level. As a result, the CDB storage structure permits flexible and efficient updates due to the different levels of granularity with which the information can be stored or retrieved
- **Layers:** The CDB model is also logically organized as distinct layers of information. The layers are notionally independent from each other (i.e., changes in one layer do not impose changes in other layers).
- **Levels-of-Detail (LODs):** The availability of LOD representations is critical to real-time performance. Most simulation client-devices can readily take advantage of an LOD structure because, in many cases, less detail/information is necessary at increasing distances from the simulated own-ship. As a result, many client-devices can reduce (by 100-fold or more) the required bandwidth to access environmental data in real-time. The availability of levels-of-detail permits client-devices to deal with data stores having big-data levels of content. The CDB storage model supports a LOD hierarchy consisting of up to 34 LODs. The CDB standard requires that each geographic area be reduced into a LOD hierarchy in accordance to the availability of source data.

The standard does not define or enforce an operating system or file system.

1.6.4 What the CDB Standard is Not

The representation and sharing of geospatial data plays a key role in the interoperation of systems and applications that use such data. In the mid to late 90's some specifications/standards were conceived to provide a means of sharing synthetic environment data, in source form, for a wide variety of applications. They provided a standardized means to share native databases, thereby avoiding direct and (often inefficient) conversion of the data to/from (often proprietary) native database format.

The CDB standard defines a model for data representation and attribution of terrain, objects and entities within the CDB data store. However, the standard does not define requirements for the movement, change in shape, physical properties and/or behavior of such objects and entities. These capabilities fall under the domain of simulation software or application.

The CDB standard does not define requirements for the representations of celestial bodies, such as the Sun, Moon, stars, and planets. Rather, this standard assumes that the modeled representation of celestial bodies is internally held within the appropriate simulator client-devices.

The CDB standard does not specify a mandatory "coverage completeness requirement" imposed by. This permits the generation of a CDB structured data store even when there is limited data availability.

Given the requirement that the CDB standard be platform independent, the standard does not provide the implementation details of specific off-line data store compilers or runtime publishers attached to specific client-devices⁹. Furthermore, since there is no standardization of the SE representation internal to client-devices (they vary by type¹⁰, by vendors), it is unlikely that such information would completely satisfy the interests of all developers. More importantly, the structure and format of data ingested by each client-device is typically proprietary. Finally, this standard does not describe the effort required to develop CDB off-line compilers and/or CDB runtime publishers.

1.7 General CDB Data Store and Implementation requirements

This section details the requirements for a CDB conformant or structured data store.

Requirement 1
http://www.opengis.net/spec/CDB/1.0/core/model
<i>A CDB implementation SHALL include all elements of the CDB Core Data Model as defined in Annex A.</i>

1.7.1 Platform Requirements

Requirements Class - Platform requirements.	
/req/core/platform	
Target type	Operations
Dependency	Operating System

⁹ Example client devices are: UAV simulators, mission planning simulators, , helicopter training simulators, and so
fort

¹⁰ It is precisely the intent of the CDB standard to provide such standardization.

Dependency	Hardware
Dependency	File Management system
Requirement 5	/req/core/literal-case

1.7.1.1 File System

Moved to section 5.1 Volume 10: Implementation Guidance

1.7.1.2 Operating System

Moved to section 5.2 Volume 10: Implementation Guidance

1.7.1.3 Transport Protocols

The CDB standard is transport protocol-independent. The standard does not mandate the use of specific transport protocols. Furthermore, this standard does not explicitly rely on or specify any transport protocols.

1.7.1.4 System Hardware Independence

Moved to section 5.3 Volume 10, Implementation Guidance

-

1.7.1.5 Literal Case

Requirement 5
http://opengis.net/spec/CDB/1.0/core/literal-case
Implementations SHALL support the literal case rules as specified in this standard. CDB file naming conventions are case sensitive. Further, regardless of case, any name such as “house” SHALL have the same semantic meaning.

CDB structured data stores may be implemented on any modern operating systems whether they are Windows-like or Unix-like.

Throughout this standard, the reader will notice that filenames and directory paths are specified using a mix of upper and lower cases. This choice is made to improve and ease the readability of those names.

However, it is important to note that no two names are to differ only by their case. After all, a name is used to designate a single object or concept whether that name is spelled in lowercase or uppercase or even using mixed case. For instance, the terms house, House, and HOUSE (and even HoUsE) all convey the same idea of a residence where people live. And this stays true for all combination of cases.

1.7.2 General Data Representation Requirements

The following is the Requirements Class for general data representation requirements.

Requirements Class - General Data Representation. (6-10)	
/req/core/data-representation	
Target type	Operations
Dependency	WGS-84 definition
Dependency	Compression Algorithms
Dependency	Units of measure
Requirement 6	/req/core/raster-imagery-compression
Requirement 7	/req/core/uom
Requirement 8	/req/core/crs
Requirement 9	/req/core/crs-client
Requirement 10	/req/core/coordinates

1.7.2.1 Compression of Storage Intensive Imagery Datasets

Requirement 6
http://opengis.net/spec/CDB/1.0/core/raster-imagery-compression
JPEG 2000 <i>SHALL</i> be used for storing and compressing Raster Imagery such as the imagery used for Out The Window (OTW) applications. See Annex C Volume 2 for reasons for this requirement.

1.7.2.2 Compression of other imagery datasets

In general, all TIFF/GeoTIFF files benefit from LZW compression. For this reason, and as a general practice, the compression of all TIFF-based raster datasets is recommended. The one exception is when every cell in the raster dataset is a unique floating point number. In this case, the compressed file may be larger than the original.

1.7.2.3 Units of Measure

Requirement 7

<http://opengis.net/spec/CDB/1.0/core/uom>

All units of measure in a CDB conformant data store *SHALL* be in meters.

1.7.3 Coordinate Reference Systems

Requirement 8

<http://opengis.net/spec/CDB/1.0/core/crs>

Geographic locations in CDB *SHALL* be expressed using WGS-84 (World Geodetic System 1984), equivalent to EPSG (European Petroleum Survey Group) code 4326 (2 dimensions) and EPSG code 4979 (3 dimensions).

If a geographic location also has an altitude, the altitude *SHALL* be expressed relative to the WGS-84 reference ellipsoid.

Please see the Volume 8: OGC CDB Spatial Reference System Guidance.

Requirement 9

<http://opengis.net/spec/CDB/1.0/core/crs-client>

Each implementation of the simulator client-devices accessing the CDB geospatial data *SHALL* at a minimum support the WGS-84 geodetic coordinate system as specified in [Req 8](#). Other reference systems may be used in the client application.

1.7.4 Geographic Coordinates

Requirement 10

<http://opengis.net/spec/CDB/1.0/core/coordinates>

Coordinates *SHALL* be described using the decimal degree format without the “°” symbol. The values of latitude and longitude *SHALL* be bounded by $\pm 90^\circ$ and $\pm 180^\circ$ respectively. Positive latitudes are north of the equator, negative latitudes are south of the equator. Positive longitudes are east of the Prime Meridian; negative longitudes are west of the Prime Meridian. Latitude and longitude are expressed in that sequence, namely latitude before longitude.

2. CDB Concepts

This chapter presents basic CDB data store model concepts. These concepts are either reused by other concepts or used repeatedly throughout the Standard.

The CDB core data model may be viewed as an instance of a Discrete Global Grid System as defined in OGC 15-104. Please note, however, that the CDB data model and structure predates the OGC DGGS activity by over a decade and as such should not be deemed compliant with the OGC DGGS standard..

A DGGS is a spatial reference system that uses a hierarchical tessellation of cells to partition and address the globe. DGGS are characterized by the properties of their cell structure, geo-encoding, quantization strategy and associated mathematical functions.

The following sections detail the CDB tiling storage model.

Requirements Class - Tiles/Geocells and LoD relationships (11-16 and 41)

</req/core/cdb-structure-tiles-lod>

Target type	Operations
Dependency	WGS-84 2d definition
Requirement 11	/req/core/geocell-extent
Requirement 12	/req/core/geocell-length
Requirement 13	/req/core/tile-sizes
Requirement 14	/req/core/lod-area-coverage
Requirement 15	/req/core/hierarchy
Requirement 16	/req/core/tile-lod-replacement
Requirement 41	/req/core/lod-organization-resolution (section 3.3.6)

2.1 Characteristics of the CDB tiling storage model

The CDB tiling approach has the following characteristics:

1. The earth model is divided (in latitude) into slices.
2. The slice's x-axis is aligned to WGS-84 lines of latitude.
3. The slice's y-axis is aligned to WGS-84 lines of longitude.
4. The number of units along the slice's y-axis for a given level of detail is the same for all slices. The earth surface geodetic dimension in arc-second of y-axis units within an earth slice is exactly the same, regardless of latitude.
5. The geodetic dimension of an x-axis unit in arc-second is constant within a zone, but is re-defined at pre-selected latitudes to achieve a greater level of spatial sampling uniformity in all tiles; this overcomes the narrowing effect of increased latitudes on longitudinal distances¹¹.
6. The number of units along the slice's x-axis for a given level of detail is the same within each zone.
7. The number of units along the slice's y-axis is constrained to a multiple of 2n in all slices.
8. The number of units along the slice's x-axis will vary depending on which zone the latitude of the slice belongs. For instance, in latitude zone 5, which goes from -50° to 50° of latitude, a CDB Geocell is $1^{\circ} \times 1^{\circ}$, in zone 4 and 6 which goes from latitude 50° to 70° the cell size is $1^{\circ} \times 2^{\circ}$. The main reason to introduce this concept is to maintain a reasonable eccentricity between the sides by trying to keep them as close to a square as possible. Variable CDB Geocell size reduces the simulator client-device processing overheads associated with the switching from one zone to another (due to small number of zones across the earth), reduces the variation of longitudinal dimensions (in meters) to

¹¹ Note: The definition of zones in the CDB is the same as those in DTED (with the exception of the poles).

a maximum of 50% and improves storage efficiency. Two requirements as defined below are used to define the size of a CDB Geocell

Geocell extent

Requirement 11
http://opengis.net/spec/CDB/1.0/core/geocell-extent
A CDB Geocell <i>SHALL</i> start and end on a whole degree along the longitudinal axis.

Geocell length

Requirement 12
http://opengis.net/spec/CDB/1.0/core/geocell-length
The length of the CDB Geocell <i>SHALL</i> be a whole factor of 180, in other words a length of 1, 2, 3, 4, 6 and 12 degrees are legal but lengths of 7 and 8 degrees would not be since these are not exact factors of 180.

2.1.1 Details of the Tiling System in the CDB core model

The CDB storage model represents the earth as a fixed number of slices divided equally along a latitude axis as illustrated in Figure 2-1: CDB Earth Slice Representation.

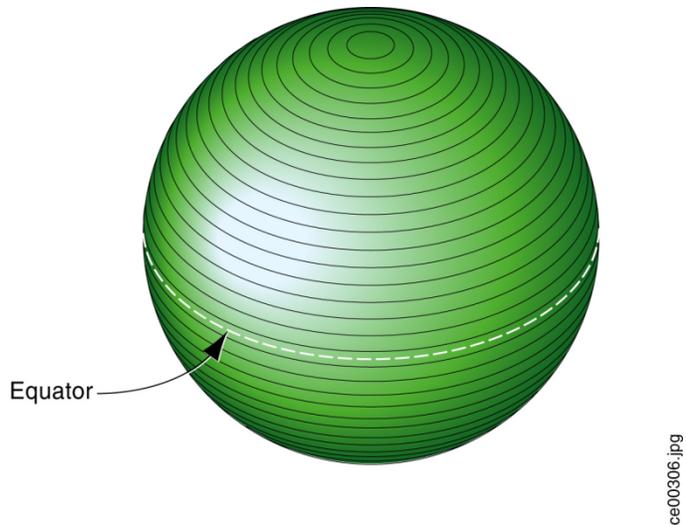


Figure 2- 1: CDB Earth Slice Representation

The earth surface coordinate system conventions used for each slice consists of a regular two-dimensional grid where the x -axis is always pointing east, aligned to WGS-84 lines of longitude and where the y -axis is always pointing north, aligned with WGS-84 lines of latitude. The earth surface origin, reference point (lat:0, long:0) on the CDB earth representation, is defined by the intersection of the WGS-84 equator and the WGS-84 international 0° meridian¹². The $x=0$ and $y=0$ reference is at (lat:0, long:0) x is positive going East and negative going West; y is positive North of the equator and negative South.

Every x and y unit corresponds to a fixed increment of longitude and latitude in arc-seconds. The x -axis and y -axis fixed increment units are hereafter referred to as $XUnit$ and $YUnit$. Since the y -axis of the slices is aligned with WGS-84 lines of longitude, y -axis coordinate units are uniformly distributed between the equator and the poles in both geodetic system terms (arc-second) and in Cartesian system terms (meters). This property naturally leads to defining the same number of $YUnit$ per slice. This however is not the case with the x -axis. As illustrated in Figure 2-2: Variation of Longitudinal Dimensions versus Latitude, the Cartesian space distance between such x -axis values diminishes as we move towards the poles. In order to maintain size constancy, the CDB standard provides a piecewise solution similar to that used by NGA DTED data. The world is divided into eleven zones. All CDB Geocells within a slice are one degree of latitude high (the height of a slice) and of varying width in longitude depending on the zone to which they belong.

¹² Specifically, the WGS 84 meridian of zero longitude is the IERS Reference Meridian, 5.31 arc seconds or 102.5 metres (336.3 ft) east of the Greenwich meridian at the latitude of the Royal Observatory.

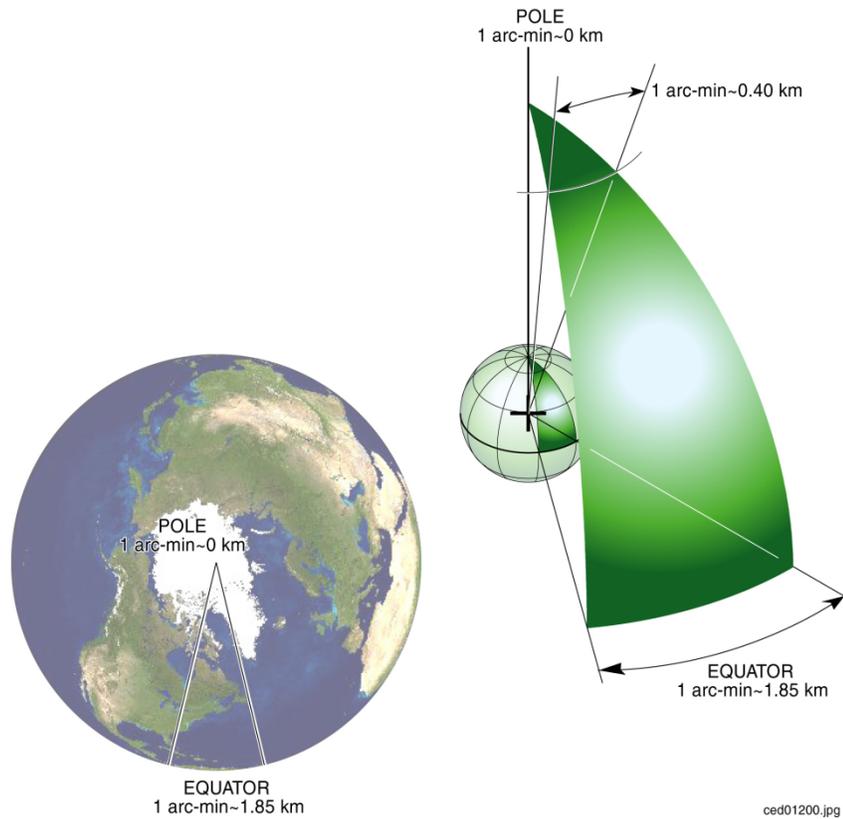


Figure 2- 2: Variation of Longitudinal Dimensions version Latitude

In order to meet one of the previously mentioned real-time considerations, the number of y-axis units for one Geocell, $NbYUnitInCDBGeocell$, is set to a power of two. This has been chosen as 1024 to give a north-south grid post spacing of approximately 109 meters at the default Level of Detail (LOD 0); this spacing is the same for all earth slices.

$$NbYUnitInCDBGeocell=1024 \quad (2-1)$$

The CDB standard also imposes an integer number of slices along latitude lines. $NbEarthSlice$ is the number of earth slice from South Pole to North Pole and is equal to 180 since each side is one degree.

$$NbEarthSlice = 180 \quad (2-2)$$

Furthermore, the number of x-axis units, $NbXUnitInCDBGeocell$, is also maintained to be the same as that of $NbYUnitInCDBGeocell$ for all CDB Geocells. As previously stated, the cell width in longitude is adjusted at specific latitudes to maintain a reasonable aspect ratio. As a consequence the area defined by the corner coordinates (x,y) , $(x+1, y)$, $(x, y+1)$, $(x+1, y+1)$, decreases when moving toward the poles in the same zone and increases when moving toward the equator.

$$NbXUnitInCDBGeocell=NbYUnitInCDBGeocell \quad (2-3)$$

The geodetic dimension of a *YUnit* is referred to as *ArcSecLatUnitInCDBGeocell*; it is the same for all slices and is determined by Equation (2-4).

$$\begin{aligned}
 \text{ArcSecLatUnitInCDBGeoCell} &= \frac{180 \text{ degrees} \times 3600 \text{ arcsec/degree}}{NbYUnitInCDBGeoCell \times NbEarthSlice} \\
 \text{ArcSecLatUnitInCDBGeoCell} &= \frac{180 \text{ degrees} \times 3600 \text{ arcsec/degree}}{1024 \frac{\text{unit}}{\text{slice}} \times 180 \text{ slices}} \\
 \text{ArcSecLatUnitInCDBGeoCell} &= 3.515625 \text{ arcsec/unit}
 \end{aligned}
 \tag{2-4}$$

ArcSecLatUnitInCDBGeocell is a constant defined by the CDB earth model and cannot be set to any other value.

Similarly, the geodetic dimension of a *XUnit* is referred to as *ArcSecLongUnitInCDBGeocell*; it varies at specific latitudes and is shown in Table 2-3: CDB Geocell Unit Size in Arc Seconds. As shown in the table, maintaining the *NbXUnitInCDBGeocell* constant causes abrupt changes in *ArcSecLongUnitInCDBGeocell* at specific latitudes. This is done, however, to achieve our objective of maintaining a reasonable aspect ratio across the earth model.

Table 2-3: CDB Geocell Unit Size in Arc Seconds

Zone	CDB Geocell size (° Lat × ° Lon)	ArcSecLatUnit InCDBGeocell	ArcSecLongUnit InCDBGeocell
0	1 × 12	3.515625	42.187500
1	1 × 6	3.515625	21.093750
2	1 × 4	3.515625	14.062500
3	1 × 3	3.515625	10.546875
4	1 × 2	3.515625	7.031250
5	1 × 1	3.515625	3.515625
6	1 × 2	3.515625	7.031250
7	1 × 3	3.515625	10.546875
8	1 × 4	3.515625	14.062500
9	1 × 6	3.515625	21.093750
10	1 × 12	3.515625	42.187500

2.1.2 Tile Levels of Detail (Tile LoD)

Since *NbXUnitInCDBGeocell* and *NbYUnitInCDBGeocell* are defined as being the same and since *NbYUnitInCDBGeocell* is constrained to a power of two, the CDB tile representation can readily reference square areas at a specified level-of-detail. These areas are delimited by longitude and latitude extents. By convention, LOD 0 always corresponds to the earth slice size of *NbXUnitInCDBGeocell* × *NbYUnitInCDBGeocell* with a Cartesian unit spacing in the range of one hundred meters at the slice’s zones boundaries and at the equator.

Numerically increasing levels of LOD (e.g., 1, 2, 3) correspond to tile datasets with progressively finer resolution (smaller spatial sampling intervals).

The x-axis and y-axis fixed increment unit per LOD, $XUnit_{LOD}$ and $YUnit_{LOD}$, are given per Equation (2–5).

$$\begin{aligned}
 XUnit_{LOD} &= \frac{XUnit}{2^{LOD}} \\
 YUnit_{LOD} &= \frac{YUnit}{2^{LOD}}
 \end{aligned}
 \tag{2-5}$$

Similarly, the number of units in the x-axis and y-axis and the total number of units in a CDB geocell, respectively defined by $NbXUnitInCDBGeocell_{LOD}$, $NbYUnitInCDBGeocell_{LOD}$, and $TotalNbUnitInSlice_{LOD}$, are computed by Equation (2–6).

$$\begin{aligned}
 NbXUnitInCDBGeocell_{LOD} &= NbXUnitInCDBGeocell \times 2^{LOD} \\
 NbYUnitInCDBGeocell_{LOD} &= NbYUnitInCDBGeocell \times 2^{LOD} \\
 TotalNbUnitInCDBGeocell_{LOD} &= NbXUnitInCDBGeocell_{LOD} \\
 &\quad \times NbYUnitInCDBGeocell_{LOD}
 \end{aligned}
 \tag{2-6}$$

Requirement 13
http://opengis.net/spec/CDB/1.0/core/tile-sizes
Tile sizes <i>SHALL</i> be 1024 × 1024 (e.g., 1024 $XUnit_{LOD}$ by 1024 $YUnit_{LOD}$)..

Thus, for positive LODs, every tile quadruples its geographic area coverage as the LOD decreases. Since each earth slice is limited to $NbYUnitInCDBGeocell$ (or 111319 m), tiles at LOD 0 have the same height as the height of an earth slice. For negative LODs, the same tile size is maintained. This imposes that the number of units in both x-axes and y-axes are recursively divided by two for every subsequent level until the total number of unit reaches one by one unit. LOD –10 is the coarsest LOD represented by a CDB slice. The finest available LOD number for a CDB structured data store is 23. Table 2-4 presents the complete list of CDB

LODs with the corresponding grid size, tile size, and the resulting approximate grid spacing at the equator.

Table 2-4: CDB LOD versus Tile and Grid Size

CDB LOD	Grid Size (n × n)	Approximate Tile Edge Size (meters)	Approximate Grid Spacing (meters)
-10	1	$1.11319 \times 10^{+05}$	$1.11319 \times 10^{+05}$
-9	2	$1.11319 \times 10^{+05}$	$5.56595 \times 10^{+04}$
-8	4	$1.11319 \times 10^{+05}$	$2.78298 \times 10^{+04}$
-7	8	$1.11319 \times 10^{+05}$	$1.39149 \times 10^{+04}$
-6	16	$1.11319 \times 10^{+05}$	$6.95744 \times 10^{+03}$
-5	32	$1.11319 \times 10^{+05}$	$3.47872 \times 10^{+03}$
-4	64	$1.11319 \times 10^{+05}$	$1.73936 \times 10^{+03}$
-3	128	$1.11319 \times 10^{+05}$	$8.69680 \times 10^{+02}$
-2	256	$1.11319 \times 10^{+05}$	$4.34840 \times 10^{+02}$
-1	512	$1.11319 \times 10^{+05}$	$2.17420 \times 10^{+02}$
0	1024	$1.11319 \times 10^{+05}$	$1.08710 \times 10^{+02}$
1	1024	$5.56595 \times 10^{+04}$	$5.43550 \times 10^{+01}$
2	1024	$2.78298 \times 10^{+04}$	$2.71775 \times 10^{+01}$
3	1024	$1.39149 \times 10^{+04}$	$1.35887 \times 10^{+01}$
4	1024	$6.95744 \times 10^{+03}$	$6.79437 \times 10^{+00}$
5	1024	$3.47872 \times 10^{+03}$	$3.39719 \times 10^{+00}$
6	1024	$1.73936 \times 10^{+03}$	$1.69859 \times 10^{+00}$
7	1024	$8.69680 \times 10^{+02}$	8.49297×10^{-01}
8	1024	$4.34840 \times 10^{+02}$	4.24648×10^{-01}
9	1024	$2.17420 \times 10^{+02}$	2.12324×10^{-01}
10	1024	$1.08710 \times 10^{+02}$	1.06162×10^{-01}
11	1024	$5.43550 \times 10^{+01}$	5.30810×10^{-02}
12	1024	$2.71775 \times 10^{+01}$	2.65405×10^{-02}
13	1024	$1.35887 \times 10^{+01}$	1.32703×10^{-02}
14	1024	$6.79437 \times 10^{+00}$	6.63513×10^{-03}
15	1024	$3.39719 \times 10^{+00}$	3.31756×10^{-03}
16	1024	$1.69859 \times 10^{+00}$	1.65878×10^{-03}
17	1024	8.49297×10^{-01}	8.29391×10^{-04}
18	1024	4.24648×10^{-01}	4.14696×10^{-04}
19	1024	2.12324×10^{-01}	2.07348×10^{-04}
20	1024	1.06162×10^{-01}	1.03674×10^{-04}
21	1024	5.30810×10^{-02}	5.18369×10^{-05}
22	1024	2.65405×10^{-02}	2.59185×10^{-05}
23	1024	1.32703×10^{-02}	1.29592×10^{-05}

As a result, at LOD -10 , a tile covers an area of approximately $111 \text{ km} \times 111 \text{ km}$ and is represented by a single grid element. At the opposite end of the table, at LOD 23 , a tile covers a minuscule area of $13 \text{ mm} \times 13 \text{ mm}$ with a corresponding grid spacing of about $13 \text{ }\mu\text{m}$.

Note the line corresponding to LOD 0 ; it highlights the point where the grid size stops increasing while the tile size starts decreasing. Figure 2-3 illustrates the hierarchy of CDB Tile LODs.

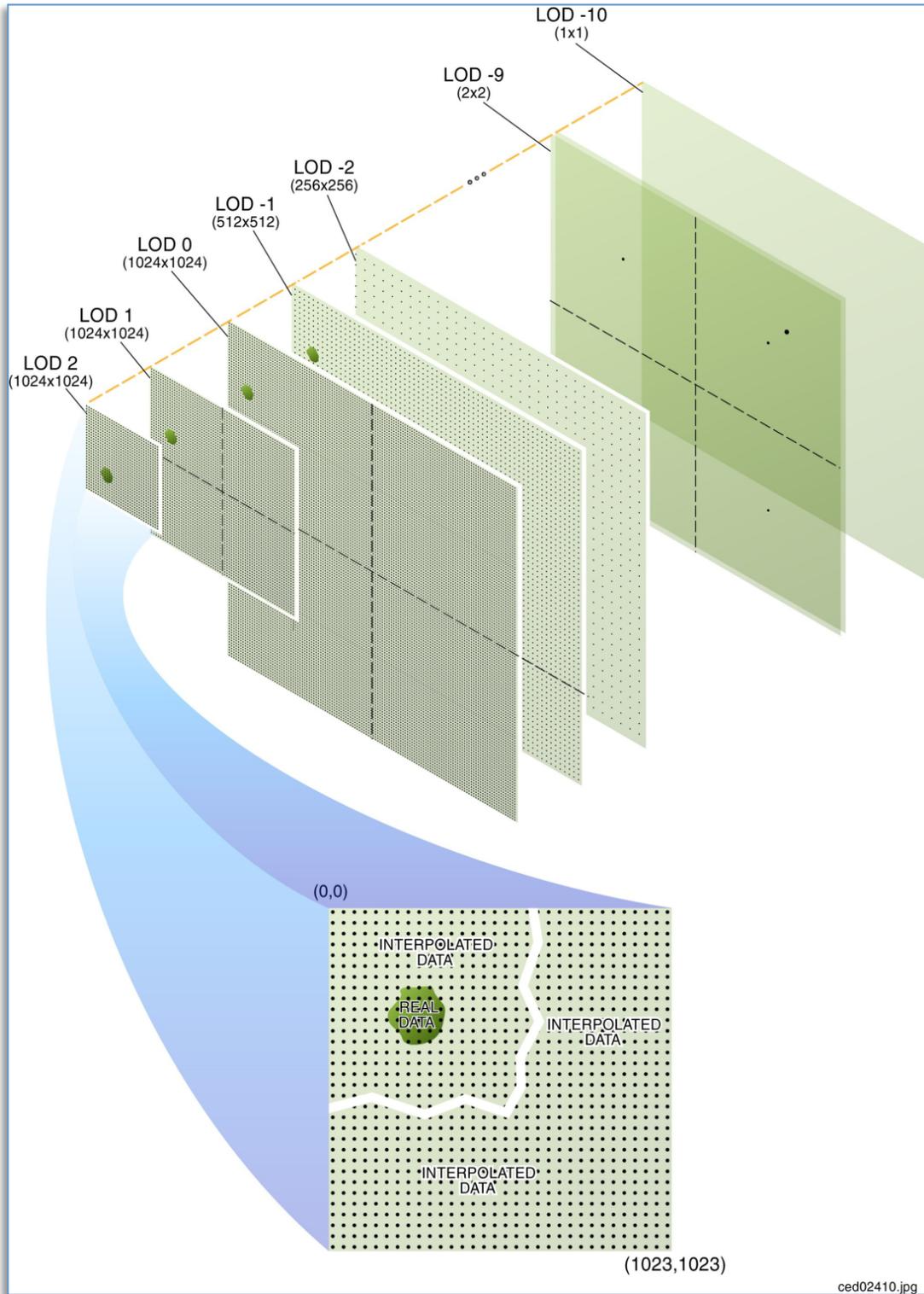


Figure 2- 3: Tile-LOD Hierarchy

2.1.2.1 Tile LoD Area Coverage Rules

Requirement 14

<http://opengis.net/spec/CDB/1.0/core/lod-area-coverage>

A tile from LOD –10 to LOD 0 *SHALL* occupy the area of exactly one CDB Geocell. This is true for all CDB Geocells from all CDB Zones. Starting with LOD 1 and up, this area *SHALL* recursively be subdivided into smaller tiles, every level corresponding to a finer representation of the previous level allowing for multiple levels of detail.

Consequently, tiles at a given LOD never overlap with others of the same LOD, and are always aligned with at least two of the edges of tiles at the previous LOD.

Figure 2-4 illustrates how different Levels of Detail (LOD's) can be combined within individual geocells. In the illustration, some CDB Geocells of the pictured earth slice are represented using five LODs, while others have only three or four LODs. Each Geocell included in a CDB data store shall be represented by an instance of at least the coarsest tile supported, i.e., one tile at LOD –10. In addition, it is not necessary that the entire geocell be represented at the same level of resolution across the entire cell. However, if a tile is present at LOD n , it implies that a coarser tile exists at LOD $n-1$ covering the area of the tile at LOD n , until LOD –10 is reached.

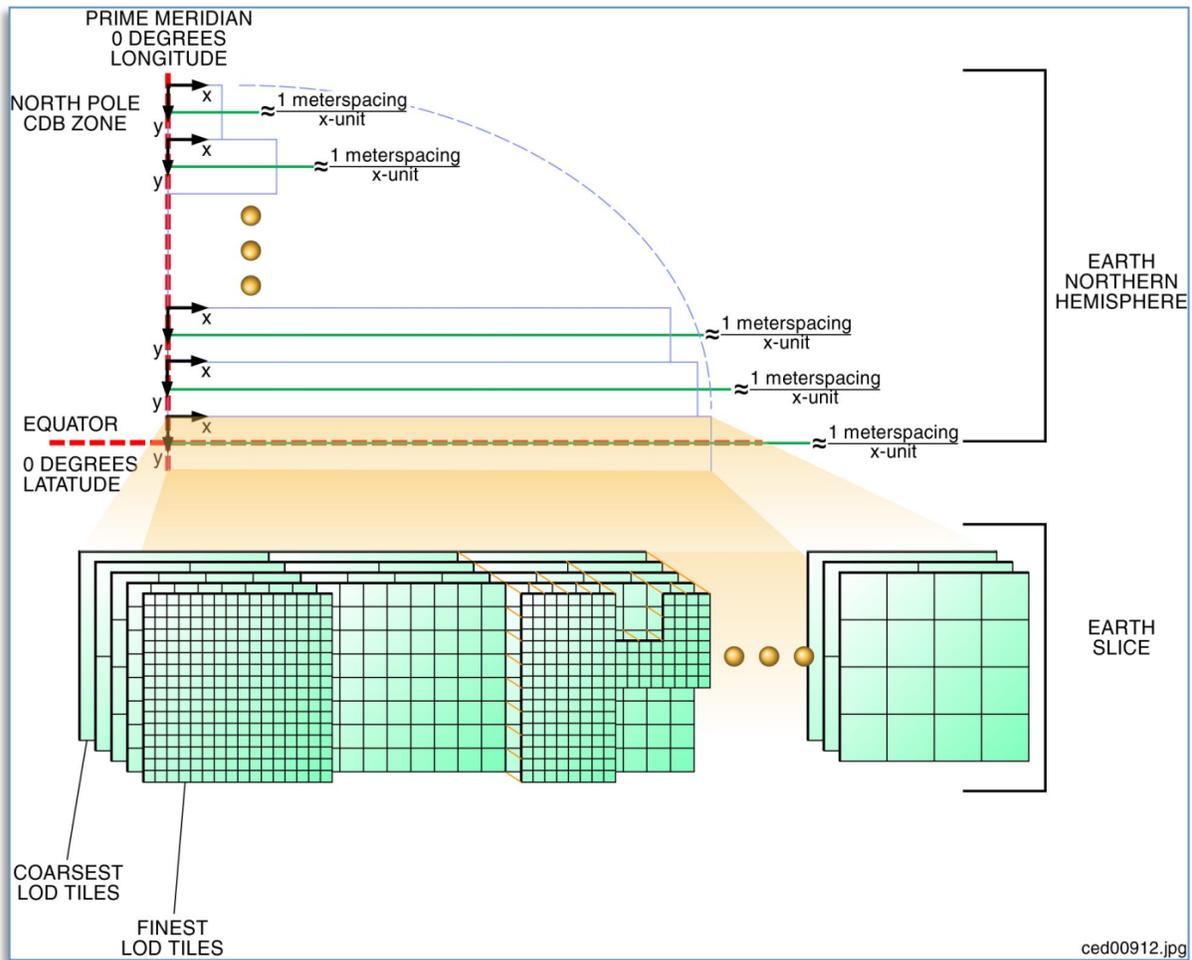


Figure 2- 4: Earth Slice Example (Five Levels of Detail)

2.1.2.2 Tile-LOD Hierarchy Rules

Requirement 15

<http://opengis.net/spec/CDB/1.0/core/hierarchy>

The LOD hierarchy of all tiled datasets *SHALL* be complete for every CDB Geocell. However, each CDB Geocell may have a different number of Tile-LODs.

2.1.2.3 Tile-LOD Replacement Rules

In general, finer tiles replace coarser tiles. The requirements are:

Requirement 16

<http://opengis.net/spec/CDB/1.0/core/tile-lod-replacement>

For negative levels of details, a tile at LOD n *SHALL* replace exactly one tile at LOD $n-1$ since both tiles cover the same area.

For positive levels of details, a tile at LOD n *SHALL* be replaced by 4 tiles at LOD $n+1$ since tiles from LOD $n+1$ cover only a quarter of the area covered by the tile at LOD n .

In the case of positive LODs, note that it is not necessary that all 4 tiles from LOD $n+1$ exist; as long as one of the four tiles is present, the replacement of the tile at LOD n can take place.

For instance, one tile at LOD -1 is replaced by one tile at LOD 0 which is in turn replaced by four tiles at LOD 1 . The replacement of coarser tiles with finer tiles stops when no finer tiles exist.

2.1.3 Handling of the North and South Pole

Zones 0 and 10 (South and North Pole) are processed differently than the other zones. As per Table 2-2: Size of CDB Geocell per Zone, this corresponds to an earth slice of 1×30 CDB Geocells.

As shown in Figure 2-2: Variation of Longitudinal Dimensions versus Latitude, a single CDB Geocell at the poles covers 12 degrees in longitude and 1 degree in latitude within a single slice. As a geographic position gets closer and closer to the poles in terms of latitude, fewer points are required in the data-grid. However the CDB Geocell still has a regular rectangular shape. Therefore, this implies that grid points will be progressively sampled in longitude in order to respect the grid format of the CDB.

In CDB Zone 0, the bottom edges of the 30 geocells of the zone all converge and collapse to a single point, the South Pole. However, the data that belong exactly to the South Pole is found in a single Geocell, the one whose lower left corner is at -90° of latitude and 0° of longitude. The redundant data representing the South Pole found in the other 29 geocells of zone 0 is ignored.

Similarly, in CDB Zone 10, the top edges of the 30 geocells of the zone also converge and collapse to a single point, the North Pole. Again, the data that belong exactly to the North Pole is found in a single Geocell whose lower left corner is at $+90^\circ$ of latitude and 0° of longitude. The redundant data representing the North Pole found in the other 29 geocells of zone 10 is ignored. The case of raster datasets that make use of the corner grid conventions is an exception since the CDB does not provide the means of representing data at precisely the North Pole ($+90^\circ$ of

latitude and 0° of longitude). In this case, it is recommended that client-devices use the average of the nearest grid elements in the immediate vicinity of the North Pole.

2.2 File System Requirements

See section 1.7.1.1

2.3 Light Naming

The CDB standard defines rules that unambiguously tag any modeled light point¹³ with a descriptive name. This provides client-devices with the information necessary to control all light points that have been tagged with a name that conforms to this standard.

The CDB standard provides a comprehensive set of light types, particularly well suited to the needs of Visual simulation. Light types include those found on:

- cultural features including point, lines, polygons¹⁴, and specialized airport systems
- air, land, and surface platforms
- life forms
- munitions

Each light type defined in this standard corresponds to a real-world light type. The standard provides a definition of each light type, which is representative of the light type's function rather than its characteristics. The client-devices use the light type name as an index to derive the properties and characteristics of the light. The approach is client-device independent because the (device-specific) client rendering parameters are not stored in the data store and are therefore invisible to the modeler and the data store tools. The modeler/tools need not be concerned with dozens of parameters that describe the light's properties and characteristics. The client-devices internally build and initialize a table of light properties and characteristics for their respective use. The table is nominally built at CDB data store load time and is built to match the device's inherent capabilities and level-of-fidelity.

The light point types are structured in a hierarchy that is designed to simplify the modeler's workload. Increasing levels of specialization are possible if a modeler specifies light names located in deeper levels of the light naming hierarchy, i.e., the more specialized the light, the deeper the level.

An extract from the light naming hierarchy is illustrated in Figure 2-5: Extract from Light Naming Hierarchy as an example. This portion of the light naming hierarchy concerns itself with lights used for "Line-based Cultural" light points (e.g., streets, highways). Immediately

¹³ The CDB standard does not distinguish between a light-point and a light-source. In the simulation industry, the term light-point refers to a point source of light that does not illuminate its immediate surroundings. Likewise, the term light-source refers to a point source of light capable of illuminating its immediate surroundings.

¹⁴ The original CDB specification as submitted to the OGC used the term "areal" instead of "polygon". In order to be consistent with OGC/ISO best practice, areal has been replaced with "polygon" throughout the document.

below the “Line-Based” level, the modeler can choose from a wide selection of lights such as Fluorescent_Light, Incandescent_Light, or Sodium_Light. A modeler that does not want to concern itself with the particular characteristics of highway lights may choose to tag its lights with a name that is higher up in CDB light name hierarchy. On the other hand, a modeler that has more elaborate source data and has more time at his disposal may choose to differentiate between “Multilane_Divided_Hwy” and “Highway” and/or “Sodium” and “Incandescent” lighting (further down in the CDB light name hierarchy).

Hazard	White light indicating the presence of an hazard around the airport
Flashing_Light	White hazard flashing light
Hi_Intensity_Light	White Hi-Intensity hazard light
Line-Based	Generic Line based lights (Linear features as Roads)
Fluorescent_Light	Fluorescent based Light
Incandescent_Light	Incandescent based Light
Mercury_Light	Mercury based Light
Metal_Halide_Light	Metal Halide based Light
Sodium_Light	Sodium based Light
Multilane_Divided_Hwy	Generic Multi-lane divided highway lights
Incandescent_Light	Incandescent based Light
Mercury_Light	Mercury based Light
Metal_Halide_Light	Metal Halide based Light
Sodium_Light	Sodium based Light
Median	Median divider Lights
Edge	Highway edge/sidewalk lights
Multilane_Hwy	Generic Multi-lane highway lights
Incandescent_Light	Incandescent based Light
Mercury_Light	Mercury based Light
Metal_Halide_Light	Metal Halide based Light
Sodium_Light	Sodium based Light
Median	Median divider Lights
Edge	Highway edge/sidewalk lights
Highway	Generic Single Lane Highway
Incandescent_Light	Incandescent based Light
Mercury_Light	Mercury based Light
Metal_Halide_Light	Metal Halide based Light
Sodium_Light	Sodium based Light

Figure 2- 5: Extract from Light Naming Hierarchy

Requirement 17
http://opengis.net/spec/CDB/1.0/core/light-name

The name of a light SHALL be composed as follows. Starting from the top-most level of the hierarchy, concatenate each of the names encountered with the backslash¹⁵ “\” character. Go down through hierarchy until the desired level of specificity is encountered.

Here are a few examples:

- \Light\Platform: A light suitable for use on all platforms
- \Light\Platform\Air\Aircraft_Helos\Formation_Light: A formation light for use on aircraft and helicopter platforms
- \Light\Platform\Land\Headlight\High_Beam_Light: A high-beam head-light for use on land vehicles
- \Light\Cultural\Line-based\Highway: A light suitable to depict highway lighting
- \Light\Cultural\Line-based\Highway\Incandescent_Light: A light used to depict incandescent highway lighting

2.3.1 Adding Names to the CDB Light Name Hierarchy

The hierarchy permits modelers to reference light types that are not defined by the current version of the CDB standard. This can be achieved by adhering to the following requirements and procedure.

Requirements Class Light Name Hierarchy (18-21)	
/req/core/light-name-hierarchy	
Target type	Data instance
Dependency	XML
Dependency	Light Name
Dependency	XML Schema – Part 2
Requirement 18	req/core/light-name-hierarchy/type name
	The modeler or the simulator vendor SHALL create a new light type name with its corresponding light code.
Requirement 19	req/core/light-name-hierarchy/type-name-definition

¹⁵ The CDB standard uses the backslash character as a separator for light names. In no time should the reader assume that the Specification is favoring the Windows operating system which is also using the backslash as a separator when building directory paths. Again, the backslash is simply a separator for names.

	The modeler or the simulator vendor SHALL provide a definition for the light type name.
Requirement 20	req/core/light-name-hierarchy/insert
	The modeler or the simulator vendor SHALL insert the new light type into the light name hierarch
Requirement 21	req/core/light-name-hierarchy/edit
	The modeler or the simulator vendor SHALL edit the Lights.xml metadata file to reflect the change to the Light Name Hierarchy

The modeler or simulator vendor may optionally provide values for Description, Intensity, Color, Frequency, Duty_Cycle... in the Lights_XXX.xml files. If the new entry has no values for Description, Intensity, etc, the new light type will immediately inherit all of the properties and characteristics of CDB-native light types higher up in the light hierarchy. If the new entry requires one or more of the fields stated in Section 2.3.2, Light Type Modeler Tuning, it will be assigned those characteristics.

Note that the level of rendering fidelity is a function of customer requirements and/or the vendor's capabilities.

The user may also elect to propose his new light name for inclusion into subsequent versions of the CDB standard.

Since the light type codes are global to a CDB structured data store, it is strongly recommended that none of the existing light type codes be modified when adding a new light type. Failure to do this would require a complete recompilation of the data store in order to map light point type name to their newly assigned light point type codes. For this reason, it is recommended that the CDB tools create new light type codes so that light relationships within the data store remain coherent.

2.4 Model Component Naming

The CDB standard provides the means to unambiguously tag any portion of a 3D model (moving model or cultural feature) with a descriptive name. As a result, client-devices have the information necessary to control all of the model components that have been tagged with a name that conforms to this standard.

The CDB standard provides a comprehensive set of model components, particularly well suited to the needs of simulation. Model components include those used on:

1. air platforms
2. buildings

3. land platforms
4. missile and rocket platforms
5. surface (maritime) platforms
6. pylons and posts

Each model component defined by this standard corresponds to a real-world model component. The XML file containing the CDB Model Components¹⁶ is part of the CDB standard distribution package and can be found in the following file: \CDB\Metadata\Model_Components.xml

The client-devices use the name as an index to provide the simulation software the needed control over the component in question.

Examples of model components are Cockpit, Turret, Rudder, Engine, Anchor, Flight_Deck, Tire, Landing_Gear, Chimney, etc.

Volume 6, OGC CDB Rules for Encoding Data using OpenFlight provides details on how to use one of these names to identify a particular model component.

2.4.1 Adding New Model Components

The user may propose missing model component names for inclusion into subsequent versions of the CDB standard. In the meantime, the missing name can be used to tag a specific model component and a simulation client-device can use that name to detect and control the new component.

2.5 Materials

This portion of the CDB standard deals with the handling of materials that make up the synthetic environment. The CDB standard provides a flexible means to store and represent materials found in the CDB representation of the synthetic environment.

In general, materials are inputs to production or manufacturing. They are often raw - that is unprocessed, but are sometimes processed before being used in more advanced production processes. A material represents the substance or substances out of which a thing is or can be made.

The CDB standard provides the means to represent:

- **Basic** (homogeneous) materials such as steel, aluminum, copper, sand, soil, stone, glass, concrete, wood, water, rubber. CDB materials are chosen for their relevance to simulation, in particular, thermal spectrum simulation.
- **Aggregates** or mixtures of basic materials

¹⁶ As of CDB Specification version 3.2, the list of CDB model components is no longer presented as an annex to avoid the risk of miscorrelation between the appendix and the metadata. The list is now exclusively found in the Metadata folder.

- **Composite** materials, i.e., a structured arrangement of basic materials or of aggregates which together represent a composite’s material that has:
 - A **Surface Substrate**
 - A **Primary Substrate**
 - One or more optional **Secondary Substrates**

The complete list of CDB Base Materials is part of the metadata provided with the CDB Schema Distribution Package and can be found in the following file:

\CDB\Metadata\Materials.xml

The following is the requirements class for CDB Materials

Requirements Class - Materials (22-28)	
/req/core/data-representation	
Target type	Operations
Dependency	Materials.xml
Requirement 22	/req/core/composite-materials-resolution
Requirement 23	/req/core/base-material-name
Requirement 24	/req/core/primary-substrate
Requirement 25	req/core/base-material-coverage
Requirement 26	req/core/composite-material-tag
Requirement 27	req/core/sem-base-materials
Requirement 28	req/core/generation-materials-3d

Requirement 22
http://opengis.net/spec/CDB/1.0/core/composite-materials-resolution
All references to Composite Materials <i>SHALL</i> , in the end, resolve down to one or more of the stated CDB Base Materials.

The task of determining a definitive list of material properties that would accommodate all of the above requirements for the today’s sensor types, vendor implementations and SEMs would be a significant challenge. Instead, the CDB Standard publicly defines a list of materials that can be used in a CDB structured data store. It is the responsibility of vendors to (internally) define the properties (that satisfies the sensor type) for these CDB materials. Vendors are totally free to select material properties that satisfy the fidelity, functionality and precision requirements of the SEM for the sensor type of interest. *See section 6.4.7.1, Volume 10 OGC CDB Implementation Guidance for additional details.*

2.5.1 Base Materials

A Base Material represents a basic (primitive) material such as water, vegetation, concrete, glass, steel. Each Base Material has a unique name. The components of a Base Material are listed in Table 2-6: Components of a Base Material.

Table 2-6: Components of a Base Material

Component	Description
Name *	Name used to represent a Basic Material.
Description	Describes the essential nature of the basic material represented. A typical example can also be provided in the description field
* Uniquely identifies the Base Material.	

Requirement 23
http://opengis.net/spec/CDB/1.0/core/base-material-name
<p>The Base Material name <i>SHALL</i> be unique, since it can be used as an index or search key in other tables (described in subsequent sections) of the CDB structure. The Base Material’s name <i>SHALL</i> always begin with the “BM_” string, followed by a unique arbitrary string that respects the following conventions:</p> <ul style="list-style-type: none"> ● Contains letters, digits, the underscore (_) or the hyphen (-) characters. ● The Base Material Name including its prefix string is limited to 32 characters.

The description of a material class gives the essential nature of the basic material represented. \CDB\Metadata\Materials.xml presents all of the Base Materials currently defined by the CDB Standard.

2.5.1.1 Base Material Table (BMT)

A Base Material Table (BMT) is provided for run-time access by client applications. See section 5.1.3, Base Material Table for more details on the file format.

2.5.2 Composite Materials

This section provides additional description and details regarding the layered substrate structure to Base Materials, aka Composite Materials.

Each Composite Material consists of a primary substrate component, an optional surface component and one or more optional secondary substrate components. Each of these components is in turn composed of one or more Base Materials described in the previous section. Components that are composed of two or more Base Materials are aggregates. Each Composite Material has a primary substrate as a minimum. The primary and secondary substrates can be optionally assigned a thickness (in meters). By definition, the surface substrate corresponds to the first micrometer (μm) to millimeter (mm) of a Composite Material. The surface substrate does not change the nature of the primary substrate; its purpose is to differentiate the object's primary substrate from its coating.

Each substrate is defined by a variable-size structure that references one or more Base Materials. Each Base Material is assigned a weight ranging from 1% to 100%. **The sum of the weights assigned to the Base Materials of each component SHALL sum to 100%**¹⁷. For example, a mixture aggregate of 75% sand and 25% soil, would be constructed as a Composite Material with a primary substrate component with Base Materials BM_SAND (75% weight) and BM_SOIL (25% weight). In this example, there is no surface substrate and no secondary substrates.

2.5.2.1 Composite Material Substrates

A substrate provides a means to describe the material composition of “hidden” materials located beneath (or inside) the surface of a feature. This information is not explicitly modeled using (for instance) polygons; instead it is an essential characteristic of the material that makes up the modeled feature.

Consider a seabed consisting of a silt deposit (Figure 2-6: Seabed Composite Material). Such a deposit might have a thickness of a few centimeters. In our model, it is considered too thick to be considered the surface substrate of the seabed. In fact, below this silt deposit, there can be sand with a thickness of a few dozen centimeters, followed by rock of (essentially) infinite thickness. A sonar device can use the thickness information provided by the Seabed Composite Material, to generate multiple echoes, corresponding to each substrate.

¹⁷ This is a requirement. The editor missed this requirement in Version 1 of the standard. This sentence will be restated in the next version as an official requirement.

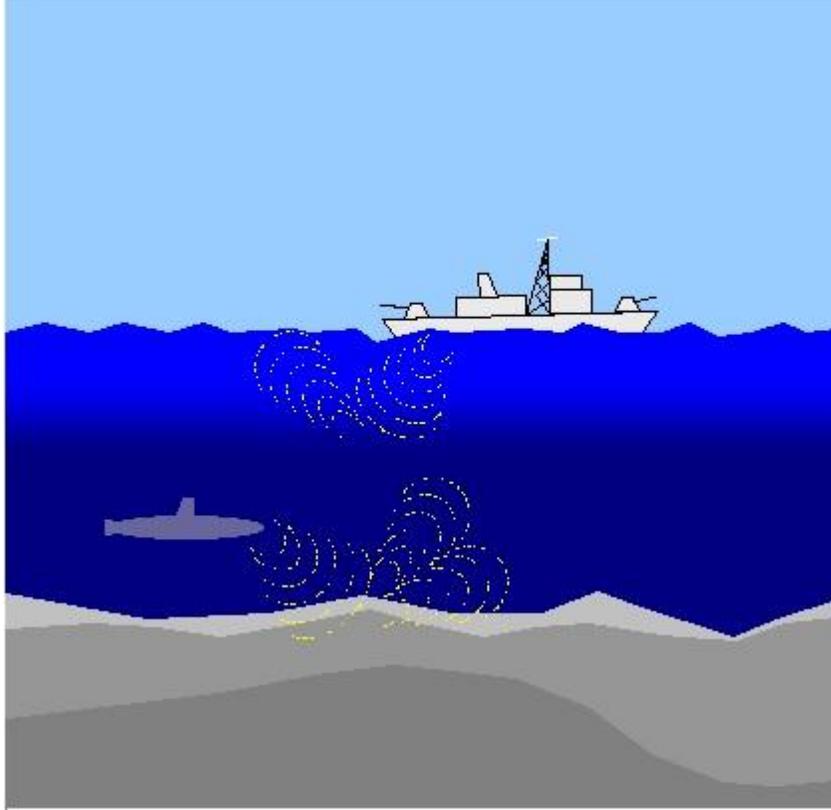


Figure 2- 6: Seabed Composite Material

As a second example, consider a half-filled refinery oil tank (see Figure 2-7: Oil Tank Composite Materials).

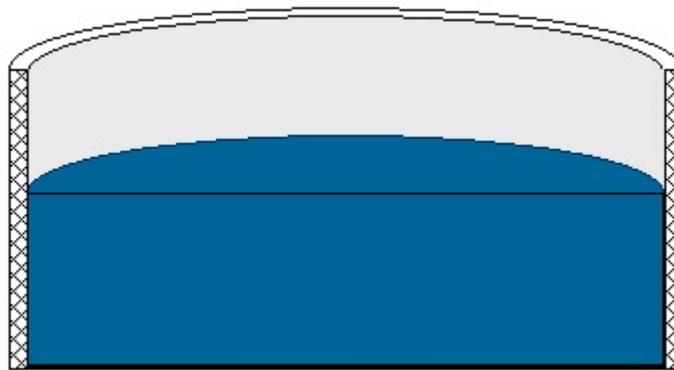


Figure 2- 7: Oil Tank Composite Materials

In order to capture different thermal signatures for the top and bottom portions of the tank, a modeler uses two different Composite Materials:

For the top half of the tank, the modeler uses a Composite Material consisting of paint (surface substrate), metal (primary substrate) and air (secondary substrate).

For the bottom half of the tank, the modeler uses a Composite Material consisting of paint (surface substrate), metal (primary substrate) and oil (secondary substrate).

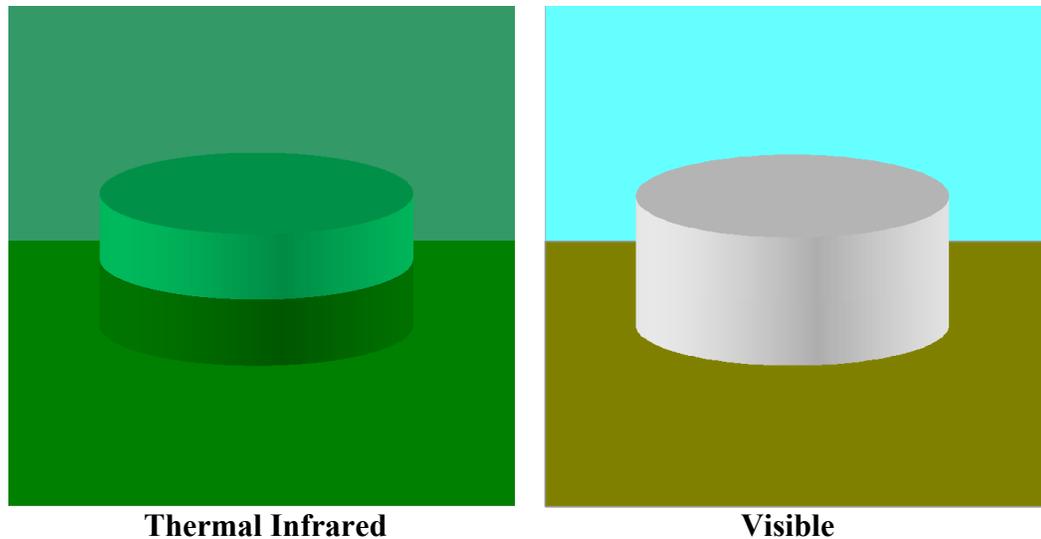


Figure 2- 8: Thermal Simulation of Oil Tank Composite Materials

Note that since the metal substrate is several centimeters thick, it is not considered to be the surface substrate of the oil. Figure 2-8: Thermal Simulation of Oil Tank Composite Materials, illustrates the different simulation responses for a FLIR and an OTW CDB client device for this particular example.

2.5.2.2 Composite Material Tables (CMT)

Composite Material Tables provide the means by which Composite Materials can be defined. Each entry within a Composite Material Table defines a structured arrangement of basic materials or of aggregates (i.e., a Composite Material). Each Composite Material entry is assigned a Composite Material Index (and an optional name). CDB datasets can then make use of the index value in order to select Composite Materials.

There are several Composite Material Tables spread across the CDB hierarchy. Note however that all Composite Material Tables follow a common XML notation that describes each Composite Material into its primary substrate, surface and secondary substrate components. Composite Materials Tables can take various forms, either as distinct XML files or embedded XML code within a file.

Here is the XML notation for a Composite Material Table:

```

<Composite_Material_Table>
  <Composite_Material index="...">
    <Name>...</Name>
    <Surface_Substrate>
      <Material>
        <Name>...</Name>
        <Weight>...</Weight>
      </Material>

      <!-- Insert other Material as needed -->

    </Surface_Substrate>

    <Primary_Substrate>
      <Material>
        <Name>...</Name>
        <Weight>...</Weight>
      </Material>

      <!-- Insert other Material as needed -->

      <Thickness>...</Thickness>
    </Primary_Substrate>

    <Secondary_Substrate>
      <Material>
        <Name>...</Name>
        <Weight>...</Weight>
      </Material>

      <!-- Insert other Material as needed -->

      <Thickness>...</Thickness>
    </Secondary_Substrate>

    <!-- Insert other Secondary_Substrate as needed -->

  </Composite_Material>

  <!-- Insert other Composite_Material as needed -->
</Composite_Material_Table>

```

Requirement 24

<http://opengis.net/spec/CDB/1.0/core/primary-substrate>

There *SHALL* be only one Primary_Substrate. If there is a Surface_Substrate there shall be only one. However, a Surface_Substrate is optional.

The Secondary_Substrate and the Thickness are optional. To specify aggregates (more than one material attribute in the MIT), the Material block is repeated. The Secondary_Substrate is provided (and optionally repeated) to described composite (stratified) materials. They appear in order starting from the Surface_Substrate, if present, followed by the Primary_Substrate (nearest to the surface), and followed by the Secondary_Substrate, if present.

Requirement 25

<http://opengis.net/spec/CDB/1.0/core/base-material-order>

The base materials that make each substrate *SHALL* be listed in decreasing order of weighting.

Requirement 26

<http://opengis.net/spec/CDB/1.0/core/composite-material-tag>

Each composite material *SHALL* be tagged with a non-negative integer index, zero being reserved for default value that is assigned by the CDB data manipulation tools.

In addition, each composite material can be optionally tagged with a descriptive name. The CDB composite material table mechanism provides the means to tag each CDB composite material with a data store tool-specific or modeler-specific composite material name.

2.5.2.3 Example 1

Consider a linear feature in a CDB data store that corresponds to a painted stripe on a runway surface. The linear feature is stored in the Man-Made Lineal dataset; the linear feature references an entry into the Geocell's Composite Material Table. That reference is the index of the Composite Material for painted asphalt. The entry pointed to describes a Composite Material whose Primary Substrate is 100% BM_ASPHALT and whose Surface Substrate is 100% BM_PAINT-ASPHALT.

2.5.2.4 Example 2

Consider a terrain polygon feature in the GSFeature dataset. The polygon feature covers a large wetland area that contains 4 Base Materials, namely BM_SOIL (21%), BM_WATER-FRESH (51%), BM_LAND-LOW_MEADOW (26%) and BM_SAND (2%). The polygon feature references an entry into the Geocell's Composite Material Table. That reference is the name of the Composite Material for wetlands. The entry describes a Composite Material whose Primary Substrate is composed of four Base Materials, namely water (with 51% weight), low height vegetation (with 26% weight), soil (with 21% weight) and sand (with 2% weight).

2.5.3 Bringing it all Together

Figure 2-9: Flow of Material Attribution Data illustrates the flow of material attribution data from features in the CDB right through to the client-device.

Each of the raster features in a CDB structured data store can (and should) reference a Composite Material. The reference points to an entry into a Composite Material Table. Each CDB tile has a

Composite Material Table. The impact of additions, deletions, and modifications to the Composite Material Table are limited to only those features that make up the tile; this reduces the compilation time associated with the production of Composite Material Table data.

Likewise, zones and polygons within a 3d model, such as OpenFlight, can optionally reference one or more Composite Materials. The references each point to entries into a Composite Material Table that is associated with the model. Each model can have an associated Composite Material Table. The impact of additions, deletions, and modifications to the Composite Material Table are limited to only those features that make up the model; this reduces the generation time associated with the production of Composite Material Table data for the model.

In turn, each of the entries in the Material Composite Table has one or more references to Base Materials entries of the Base Materials Table. The Base Materials Table is global to the CDB and its contents are defined and governed by this Standard.

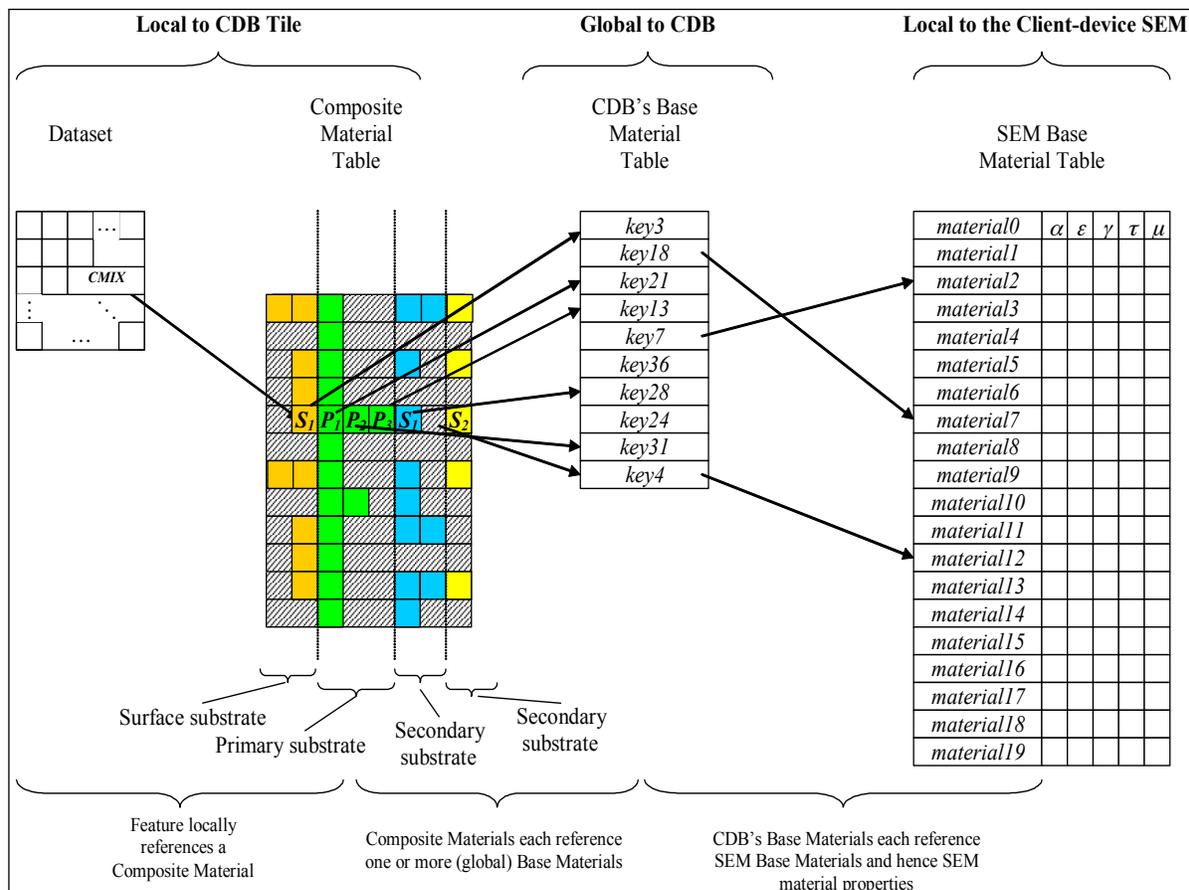


Figure 2- 9: Flow of Material Attribution Data

2.5.4 Determination of Material Properties by Sensor Environmental Model (SEM)

Please refer to implementation guidance in Volume 10 OGC CDB Implementation Guidance, Sections 6.4.7.1 and 6.4.7.2.

Requirement 27

<http://opengis.net/spec/CDB/1.0/core/sem-base-material>

The specialist *SHALL* ensure that his SEM has a corresponding Base Material for each of the CDB Base Materials.

2.5.5 Generation of Materials for Inclusion in CDB Datasets

In the case of vector data, the generation of the material information typically requires the modeler to apply an image classification process to the terrain raster imagery. Many industry-standard tools offer this classification capability.

Following this step, the resultant material classified raster imagery is vectorized into polygons (polygons) and/or lineals. Note that the quality of the image classification typically improves with the availability of multispectral terrain imagery data. Also note that these two steps can be skipped if the vectorized datasets already exist in digital form.

The classification of the terrain imagery can be done directly against the Base Materials defined by \CDB\Metadata\Materials.xml. In this case, the modeler need not be aware of the mandated Base Materials. This can be done because the tools can abstract these Base Materials and provide the modeler with an alternate selection of materials. The selection of materials provided to the modeler is quite arbitrary. This indirect step allows modelers to work with the “materials” they are familiar with. Nonetheless, the tools must, in the end, build the Composite Material Tables required by the CDB standard and resolve all material references into the Base Materials supported by this Standard. In effect, the Composite Material Table is used to map the modeler’s materials into CDB Base Materials.

Alternately, the classification of the terrain imagery can be done against whatever “material classes” modelers are accustomed to use when conducting such classifications. In this case, the SEM specialist can define corresponding Composite Materials for each of these material classes so that they resolve down to the Base Materials supported in the CDB data store.

Requirement 28

<http://opengis.net/spec/CDB/1.0/core/generation-materials-3d>

In the case of 3D models, the modeler *SHALL* appropriately tag zones or selected polygons with the appropriate materials. Here again, the modeler need not be aware of the Base Materials mandated by the CDB standard and can work with the materials he is

most familiar with.

3. CDB Structure

This chapter defines the CDB data store physical structure, i.e., the name of all directories forming the CDB model hierarchy, as well as the name of all files found in the CDB model hierarchy. An important feature of the CDB Model is the fact that all CDB file names are unique and that the filename alone is sufficient to infer the path to get to the file.

The CDB is composed of several datasets that usually reside in their own directory structure; however some datasets share a common structure. The following sections present the directory structures for all CDB conformant datasets.

3.1 Top Level CDB Model/Structure Description

The top-level directory structure of the CDB from the root directory is described below. All of the synthetic environment content falls in these directories:

1. **\CDB\:**

This is the root directory of the CDB. It does not need to be “\CDB\” and can be any valid path name on any disk device or volume under the target file system it is stored on. In order for the text of this standard to remain readable, all examples referring to the root CDB path name will start with \CDB\. A CDB cannot be stored directly in the root directory of a disk device or volume. A CDB path name cannot be within another CDB or CDB version. The length of the path name leading to the CDB root directory should be small enough such that the platform file system can store all possible file path names stored within a CDB.

Requirements Class (29-32)	
/req/core/cdb-root-requirements	
Target type	Data instance
Dependency	XML
Dependency	CDB file hierarchy
Dependency	XML Schema – Part 2

Requirement 29	req/core/root-file-hierarchy
	All of the files stored within a CDB data store <i>SHALL</i> be under the root directory or within a subdirectory under the root directory
Requirement 30	req/core/root-give-path
	Run-time applications <i>SHALL</i> be given the path and device on which the CDB is stored in order to access the CDB.
Requirement 31	req/core/root-access-version
	The CDB standard also has provisions for the handling of multiple, incremental versioning of the CDB. To support this capability, run time applications <i>SHALL</i> first access a predetermined version of the CDB and all its predecessors to determine content changes to the CDB.
Requirement 32	req/core/root-version-default
	If no change is encountered in any of the incremental versions, the applications <i>SHALL</i> use the content of the active default CDB. The versioning mechanism is done at the file level. Refer to Section 3.2, CDB Configuration Management, for details on how CDB supports incremental versioning.

2. **\CDB\Metadata**: The directory that contains the specific XML metadata files which are global to the CDB. The directory structure and metadata descriptions are defined in Section 3.1.1, Metadata Directory.
3. **\CDB\GTModel**: This is the entry directory that contains the Geotypical Models Datasets. The directory structure is as defined in Section 3.4, GTModel Library Datasets.
4. **\CDB\MMModel**: This is the entry directory that contains the Moving Models Datasets. The directory structure is defined in Section 3.5, MModel Library Datasets.
5. **\CDB\Tiles**: This is the entry directory that contains all tiles within the CDB instance. The directory structure is defined in Section 3.6, CDB Tiled Datasets.
6. **\CDB\Navigation**: This is the entry directory that contains the global Navigation datasets. The directory structure is defined in Section 3.7, Navigation Library Dataset

Most CDB datasets are organized in a tile structure and stored under **\CDB\Tiles** directory. The tile structure facilitates access to the information in real-time by the runtime client-devices. However, for some datasets such as Moving Models or geotypical models datasets that require minimal storage, there is no significant advantage to be added from such a tile structure. Such datasets are referred to as global datasets: they consist of data elements that are global to the earth, i.e., no structure other than the datasets is provided.

3.1.1 Metadata Directory

There is one directory containing metadata files that are global to the overall CDB structure. \CDB\Metadata contains metadata files that define the various sets of naming hierarchies and definitions. File content is described in Section 5.1, Metadata Datasets. Most metadata files (except one) are optional and CDB users must implement default behaviors as defined in the requirements. The \CDB\Metadata directory contains the following metadata files:

- “Lights” Definitions Metadata file: This file contains the metadata that defines the light points name hierarchy for the CDB. Refer to Section 2.3, Light Naming, for a description of the light type hierarchy. A listing of the CDB light type hierarchy can be found in Annex J. If “Lights.xml” is not found in the metadata directory then use the hierarchy found in Annex J (Volume 2 OGC CDB Core Model and Physical Structure Annexes). Refer to section 5.1.1 Light Name Hierarchy Metadata for a description of the light point name hierarchy file.
- “Model_Components” Definitions Metadata file: This file contains the metadata that defines the CDB model components. Refer to Section 2.4, Model Component Naming, for a description of the model components. The XML file containing the CDB Model Components¹⁸ is part of the CDB standard distribution package and can be found in the following file: \CDB\Metadata\Model_Components.xml. Refer to section 5.1.2 of this document - Model Components Definition Metadata - for a description of the model component file.
- “Materials” Definitions Metadata file: This file contains the base material names for the CDB. Refer to Section 2.5, Materials, for a description of the CDB materials. A listing of the CDB Base Materials can be found in \CDB\Metadata\Materials.xml. Refer to section 5.1.3 Base Material Table for a description of the materials definition file.
- “Defaults” Definitions Metadata file: This file contains the default values for each of the CDB datasets. Refer to Chapter 5, CDB Datasets, for a description of the CDB datasets. Annex S of the OGC CDB Core: Model and Physical Structure: Informative Annexes lists the various default values as documented throughout this standard. Defaults values found in Annex S shall be used if the “Defaults.xml” file is not found in the metadata directory. Refer to section 5.1.4 Default Values Definition Metadata for a description of the defaults definition file.
- “Specification_Version” Metadata file (deprecated)
- “Version” Metadata file: This metadata file is mandatory and identifies the content of one CDB Version. The concept is described in section 3.2.1, CDB Version; the content of the file is defined in section 5.1.6, Version Metadata. See Requirement 38
- “CDB_Attributes” Metadata file: This file is used to describe all the CDB attributes that are supported by the CDB standard. A complete listing and description of CDB attributes is provided in section 5.7.1.3, CDB Attributes of this standard. The file is described in section 5.1.7 CDB Attributes Metadata.

¹⁸ As of CDB Specification version 3.2, the list of CDB model components is no longer presented as an annex to avoid the risk of miscorrelation between the appendix and the metadata. The list is now exclusively found in the Metadata folder.

- “Geomatics_Attributes” Metadata file: This file is used to describe all Geomatics attributes that may be referenced by this CDB (refer to section 5.7.1.2.6.2, Geomatics Attributes for a description of Geomatics attributes). Note that the usage of Geomatics attribution falls outside of the jurisdiction of the CDB standard. Nonetheless, the CDB standard provides a standardized mechanism to allow users to fully describe each of the Geomatics attributes they wish to insert within the CDB repository structure. The file is described in section 5.1.8, Geomatics Attributes Metadata.
- “Vendor_Attributes” Metadata file: This file is used to describe all Vendor attributes that may be referenced by this CDB (refer to section 5.7.1.2.6.3, Vendor Attributes for a description of Vendor attributes). Note that the usage of Vendor attribution falls outside of the jurisdiction of the CDB standard. Nonetheless, the CDB standard provides a standardized mechanism to allow users to fully describe each of the Vendor attributes they wish to insert within the CDB repository structure. The file is described in section 5.1.9, Vendor Attributes Metadata.
- Client-specific Metadata files: These files are limited to “**Lights_xxx.xml**” Definitions Metadata files and offer a complementary approach to modifying the appearance of lights for specific client-devices. The “xxx” suffix is a 32-character string placeholder that stands for the client-device name. There can only be one such file per client-device and the files for each client-device are optional. Refer to section 5.1.1.1, Client Specific Lights Definition Metadata for a description of the client specific lights definition file.
- “Configuration” Metadata file: This file provides the means of defining CDB Configurations. The concept is defined in section 3.2.4, CDB Configuration; the content of the file is defined in section 5.1.10, Configuration Metadata.

3.1.2 Metadata File Examples

Requirement 33
http://opengis.net/spec/CDB/1.0/core/metadata-version
Each CDB Version SHALL have a metadata file whose complete path and filename is: \CDB\Metadata\Version.xml

Requirement 34
http://opengis.net/spec/CDB/1.0/core/metadata-flir
A Forward Looking Infrared client device named “FLIR” SHALL have a client specific metadata file having the following directory path and filename: \CDB\Metadata\Lights_FLIR.xml

3.2 CDB Data Store Configuration Management

The CDB Configuration and CDB Version mechanisms allow users to manage the CDB data store. The two mechanisms permit the users to implement Configuration Management (CM) and versioning by offering the following capabilities:

- The CDB can have multiple simultaneous independent CDB Configurations.
- Each CDB Configuration is defined by an ordered list of CDB Versions.
- A CDB Version is either a collection of pure CDB Datasets or a collection of user-defined datasets in which case the CDB Version is called a CDB Extension

3.2.1 CDB Data store Version

A CDB Version is a collection of pure CDB Datasets and/or user-defined datasets. A CDB Version contains data belonging to a single version of the standard. A CDB Version may refer to another CDB Version. This is the basis for the CDB File Replacement Mechanism. The concept of a CDB Version is illustrated by the UML diagram below.

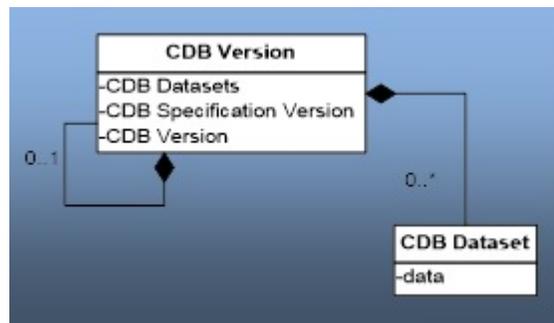


Figure 3- 1: UML Diagram of CDB Version Concept

The diagram shows that a CDB Version contains CDB Datasets; in addition it states which CDB St Number has been used to build the CDB content; finally, the CDB Version has a reference to another CDB Version. This reference allows the creation of a chain of CDB Versions. By chaining two CDB Versions together, the user can replace files in a previous CDB Version with new ones in a newer CDB Version. The figure below illustrates the chaining of CDB Versions belonging to different CDB Standard Number.



Figure 3- 2: A Valid Chain of CDB Versions

The figure above shows three (3) CDB Versions, each containing data compliant to a different version of the standard. It shows that CDB Version 3 (on the left) complies with version 3.2 of the Standard and refers (the blue line) to CDB Version 2 (in the middle), a 3.1-compliant data store, which in turn refers to CDB Version 1 (to the right), a 3.0-compliant data store.

Each CDB Version has its own Version.xml file in its Metadata folder. As such, the smallest CDB Version contains a single file: \CDB\Metadata\Version.xml

Since a CDB is made of at least one CDB Version, an empty and valid CDB has exactly one file, Version.xml, and all other datasets assume their default values.

3.2.1.1 CDB Extensions

A CDB Extension is a special CDB Version that is making use of the extension mechanism defined by this Standard to supplement the CDB with user-defined data. The actual way user-defined data is formatted and stored in a CDB Extension falls outside the realm of the Standard and is completely left to the user. The following UML diagram defines the CDB Extension concept.

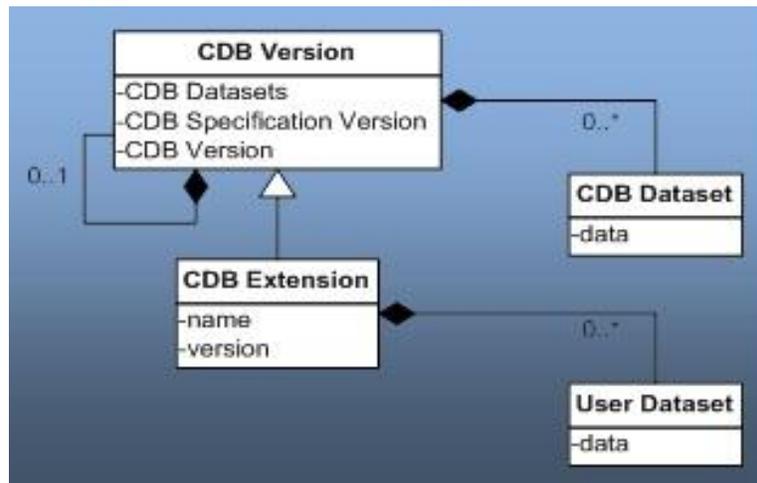


Figure 3- 3: UML Diagram of CDB Extension Concept

The diagram shows that a CDB Extension inherits all the attributes of a CDB Version and adds its own attributes, a name and a version number (of the extension). A client application checks the name attribute to recognize and process known CDB Extensions; unrecognized CDB Extensions are skipped.

To illustrate the rule, assume that CDB Version 2 from Figure 3-2 above is in fact a CDB Extension whose name is not recognized by the client application; then the client must skip CDB Version 2 and continue its processing with CDB Version 1.

3.2.2 CDB Version Directory Structure

The files and the directory structure of CDB Versions are defined in subsequent section of this chapter. There can be an arbitrary number of CDB Versions in the CDB.

Requirements Class - CDB Versioning (35-38)	
/req/core/vesioning	
Target type	Operations
Dependency	File structure
Requirement 35	/req/core/cdb-not-in-root-directory
Requirement 36	/req/core/cdb-version-path
Requirement 37	/req/core/cdb-version-entry-point-access
Requirement 38	req/core/cdb-chain-max

The root of each CDB Version can have any valid path name¹⁹ on any disk device or volume under the target file system it is stored on.

Requirement 35
http://opengis.net/spec/CDB/1.0/core/cdb-not-in-root-directory
A CDB Version <i>SHALL</i> not be stored directly in the root directory of a disk device or volume

Requirement 36
http://opengis.net/spec/CDB/1.0/core/cdb-version-path
A CDB Version path name <i>SHALL</i> not be within another CDB Version

The length of the path name leading to the CDB Version directory should be small enough such that the platform file system can store all possible file path names stored within a CDB version.

Requirement 37

¹⁹ As defined in section 2.2, File System

<http://opengis.net/spec/CDB/1.0/core/cdb-version-entry-point-access>

The path to the boot CDB Version is the entry point that *SHALL* be provided to all client-device applications when loading the CDB synthetic environment. Run-time applications *SHALL* have access, directly or indirectly, to all disk devices and volumes as well as all paths to all linked CDB Versions simultaneously.

3.2.3 CDB File Replacement Mechanism

The CDB File Replacement Mechanism allows content to be added, deleted and modified from the CDB. A file is said to exist in two (or more) CDB Versions when its relative path and name are the same in each version. This mechanism describe herein defines how to handle identical files found in multiple CDB Versions. Each CDB Version can contain a set of additions, modifications and deletions with respect to prior CDB Versions.

Figure 3-4 illustrates the case where a modeler has created a CDB Version that contains an additional level-of-detail to a wellhead model. When processed by a client application, the “effective” CDB data store now contains both the AA051_Wellhead_LOD0 and the AA051_Wellhead_LOD1 files.

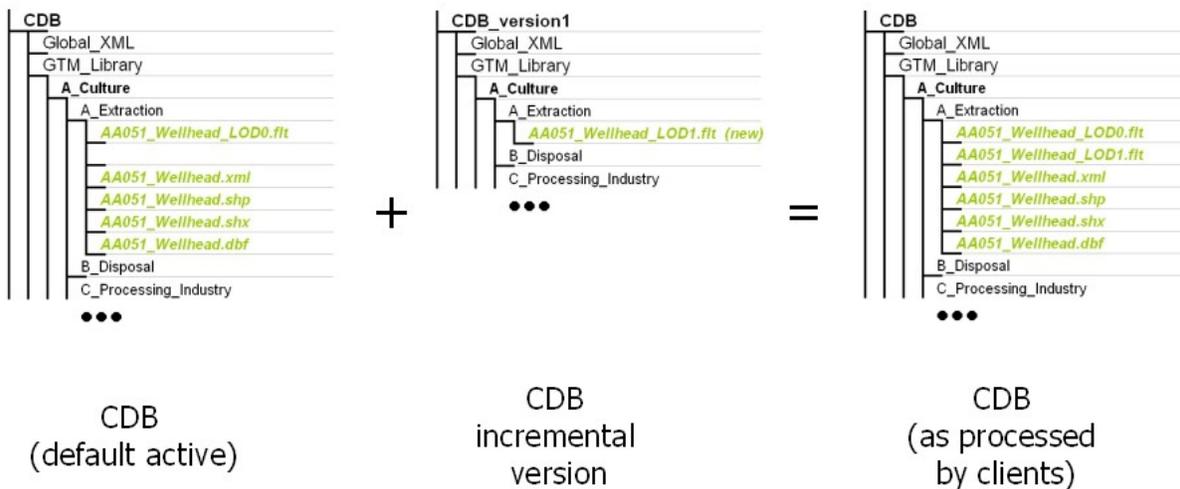


Figure 3- 4: Adding Content to the CDB data store

The process of modifying files is similar to adding files; any files that have been modified are inserted in a new CDB Version. Figure 3-5: Modifying Content of the CDB data store, illustrates the case where a modeler has modified level-of-detail #1 of a wellhead model. When processed by a client application, the “effective” CDB now contains the modified version of the AA051_Wellhead_LOD1.

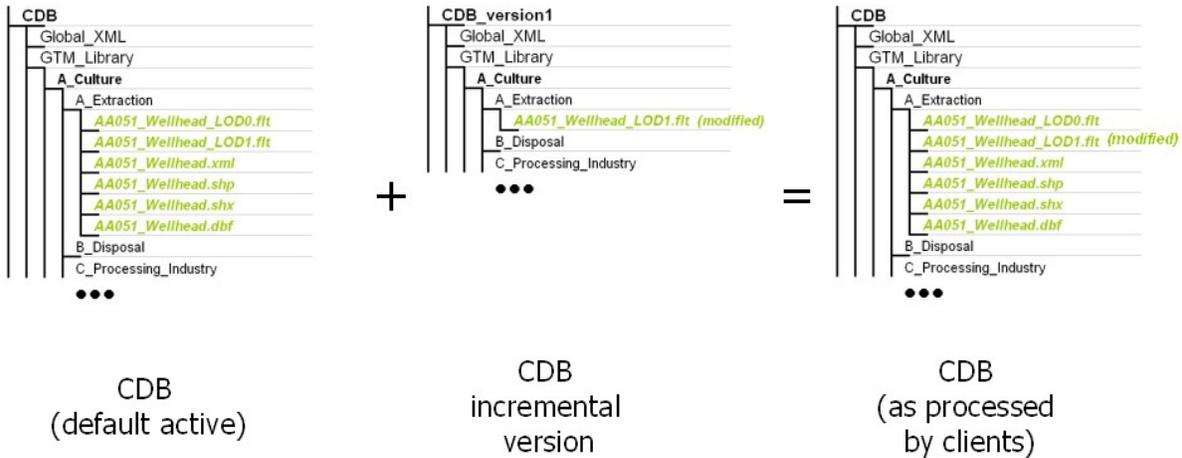


Figure 3- 5: Modifying Content of the CDB Data Store

A CDB Version can be created when content needs to be deleted from a prior CDB Version. The instruction to remove content from the CDB is triggered from the null files (e.g., files that are empty and whose size is zero) that are encountered within a CDB Version. Whenever a client application encounters a null file, it stops searching for it in prior CDB Versions and consider the file absent from the CDB. Figure 3-6: Deleting Content from the CDB, illustrates the case where a modeler has deleted level-of-detail #1 of a wellhead model. When processed by a client application, the “effective” CDB no longer contains the AA051_Wellhead_LOD1 file.

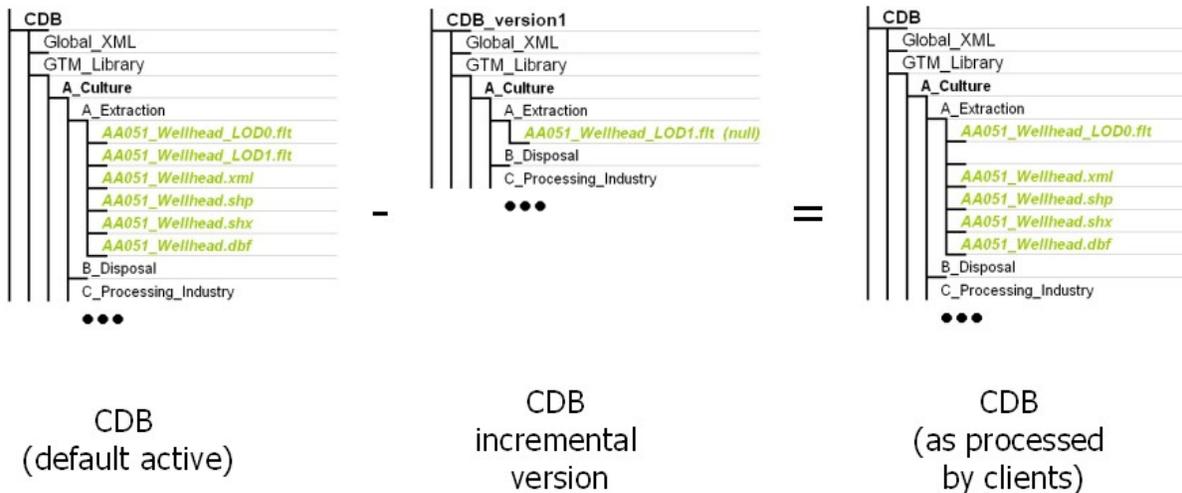


Figure 3- 6: Deleting Content from a CDB Data Store

3.2.3.1 How to handle Archives

The CDB File Replacement Mechanism works at the file level, as the name implies. For this reason, in the case of a geospecific 3D model (GSModel) datasets whose files are stored as ZIP files, the replacement is done at the ZIP level; i.e., the content of the current version of the ZIP file completely replaces the previous version.

3.2.3.2 How to handle the metadata directory

The File Replacement Mechanism does not apply to the content of the Metadata directory because the files in a CDB Version must be generated, interpreted and processed with their own metadata. Stated otherwise, the Metadata of a CDB Version belongs solely to the files residing inside that CDB Version. When generating a CDB Version, the content generation tool must also generate the Metadata that will permit a client device to consume and interpret its content. Consequently, when a client device consumes data from a particular CDB Version, the client application should retrieve and use the Metadata of that CDB Version to correctly interpret the data obtained from it.

3.2.4 CDB Configuration

A CDB Configuration defines a list of CDB Versions. The following UML diagram presents the CDB Configuration concept.

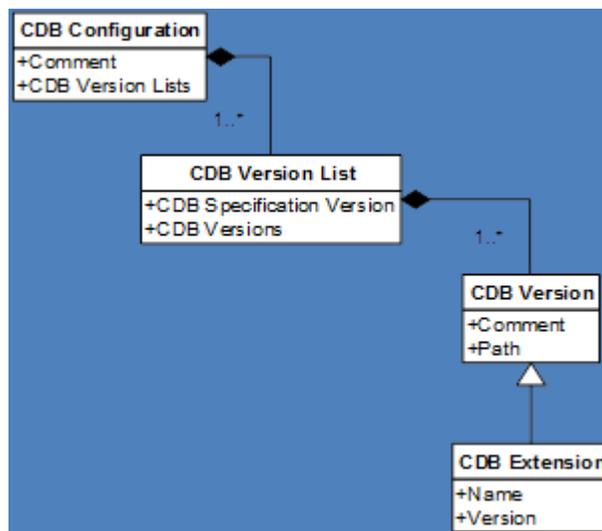


Figure 3- 7: UML Diagram of CDB Configuration Concept

The UML diagram tells us that a CDB Configuration is a collection of one to many Lists of CDB Versions. Each list of CDB Versions belongs to a single version of the CDB standard and has a collection of one to many CDB Versions. Note that a CDB Extension is a CDB Version that is making use of the extension mechanism defined by this standard to supplement the CDB with user-defined data.

The Configuration.xml metadata file provides the means of defining a CDB Configuration. The file resides in the Metadata folder of the CDB as follows:

\CDB\Metadata\Configuration.xml

When a client application opens a CDB, it searches the Metadata folder for the presence of the Configuration.xml file. If the file is found, the client uses its content to access all CDB Versions that are making up this CDB configuration. Otherwise, the client falls back to the mechanism associated with Version.xml. Note that when the client finds Configuration.xml, it does not need to open any of the Version.xml files associated with the CDB Versions referred to by the CDB Configuration; i.e., the purpose of the Configuration.xml file is to avoid reading multiple Version.xml files scattered all over the CDB.

3.2.5 Management of CDB Configurations and Versions

The performance of real-time simulation systems is directly affected by the number of CDB versions in the currently active CDB configuration. Unless the number of versions is bounded, performance guarantees cannot be provided by client-devices.

Requirement 38

<http://opengis.net/spec/CDB/1.0/core/cdb-chain-max>

Since a CDB is usually intended for use in real-time simulation systems, all CDB chains *SHALL* be limited to no more than 8 CDB versions²⁰. This requirement is meant to ensure adequate performance of an implementation using a CDB data store.

Failure to do this may result in unsuitable delays when performing simulator repositions or may lead to paging artifacts at higher speeds and/or lower-altitudes. Client-device data sheets should specify the criteria under which performance can be guaranteed for the specified training requirements.

In the case where a CDB is solely intended as an off-line (read-write) repository it is permissible to have chains with up to 50 versions. Processing times may increase with chain lengths, commensurate with storage system access times.

3.3 CDB Model Types

Requirement 39

²⁰ For instance, this would allow for configurations that consist of 1 version for a background world, 3 versions for each of the CDB specification versions, 1 version for dynamic changes, and 3 modeling content versions (1 content version for each specification version).

<http://opengis.net/spec/CDB/1.0/core/cultural-representation>

The cultural features of a CDB data store *SHALL* be represented using one of the following types of modeled representations:

- a) GTModel: 3D modeled geotypical representation of a point-feature that is anchored to the ground.
- b) GSModel: 3D modeled geospecific representation of a point-, lineal- or polygon-feature that is anchored to the ground.
- c) T2DModel: 2D modeled geospecific or geotypical representations of point-, lineal and polygon features that are anchored to the ground.
- d) MModel: 3D modeled representations of point-features that are not anchored to the ground.

The modeled representation of a feature primarily consists of its geometry and textures, and encompasses its exterior and interior.

In this Standard, the following terms and expressions are used:

- The term *Model* refers to all of the modeled representations of a cultural feature.
- The term *Model-LOD* refers to a specific level of detail of a *Model*.
- The term *2DModel* refers to the modeled representations of a 2D feature, i.e., a feature that has no significant height with respect to the underlying terrain.
- The term *2DModel-LOD* refers to a specific level of detail of a *2DModel*.
- The term *3DModel* refers to the modeled representation of a 3D feature that can be readily distinguished from the underlying terrain. In the case where the 3DModel is unique, it is referred to as a GSModel. In the case where the 3DModel is instanced, it is referred to as a GTModel. A 3DModel that is capable of movement is called a MModel. In the case where a MModel is positioned by the modeler, it is called a statically-positioned MModel.
- The term *3DModel-LOD* refers to a specific level of detail of a *3DModel*.

3.3.1 GTModel (Geotypical 3D Model)

A feature is said to have a 3D geotypical modeled representation if it is associated with a 3D Model that is typical of the feature's shape, size, textures, materials, and attributes. The use of geotypical models is appropriate if the modeler does not wish to fully replicate all of the unique characteristics (e.g., shape, size, texture) of a feature, as they are in the real-world. When a feature is represented by a geotypical model, the modeler is in effect stating that two or more features of the same type (i.e., samefeature code) have the same modeled representation.

3.3.2 GSModel (Geospecific 3D Model)

A feature is said to have a 3D geospecific modeled representation if it is associated with a 3D model that is unique in shape, size, texture, materials, and attributes. The use of geospecific

models is appropriate if the modeler wishes to fully replicate all of the unique characteristics (e.g., shape, size, texture) of a feature, as they are in the real-world. As a result, a geospecific model usually corresponds to a unique real-world recognizable cultural feature. Real-world features such as the Eiffel Tower, the Pentagon, or the CN Tower, to name a few, are usually modeled as geospecific.

3.3.3 T2DModel (Tiled 2D Model)

A feature is said to have a 2D modeled representation if it is associated with a modeled representation that has no significant height with respect to the underlying terrain and generally conforms to the terrain profile. It is convenient to think of the 2D Models as a complement and as an extension to the Primary Elevation and (VSTI) Imagery datasets. 2D Models provide the means to represent 2D surface features that are conformed to the underlying terrain:

- a) Modeled representation of geotypical and geospecific 2D lineal-features such as roads, runways and taxiways, stripes.
- b) Modeled representation of geotypical and geospecific 2D polygon-features such as aprons, surface markings, contaminants, land usage (campgrounds, farms, etc.).

2D Models can also be used to model geotypical terrain textures as a mesh of 2D textured polygons overlaying the terrain. This modeling technique replicates approaches used in early Image Generators which had limited ability to page-in geospecific terrain textures.

3.3.4 MModel (Moving 3D Model)

A moving model is typically characterized as such if it can move (on its own) or be moved. More specifically within the context of this standard, the model is not required to be attached to a cultural point feature.

During the course of a multi-player simulation²¹, each client-device is typically solicited to provide a modeled representation of each of the players. The activation of such players requires that the client-device access the appropriate modeled representation for each of players. There are a large number of military simulations where the player types are characterized by their DIS code. To this end, the CDB data store provides a moving model library whose structure provides a convenient categorization of models by their DIS code.

3.3.5 Use of GSModels and GTModels

Sections 3.3.1 and 3.3.2 illustrate cases where the choice to represent a feature with either a geotypical (GTModels) or a geospecific model (GSModels) is more clear-cut. This section gives additional insight into the considerations and tradeoffs that go with associating a point-feature with either a geotypical or a geospecific modeled representation. By characterizing a feature as geotypical, the modeler makes a statement as to the expected usage of the feature (and its associated modeled representation) within the CDB.

²¹ The players may be virtual (e.g., other simulators), synthetic (e.g., computer-generated simulations) or may be live (real-world players playing alongside virtual or synthetic players).

When a feature is tagged as geotypical...

- a) the modeler is making a statement about his knowledge of the very high probability of repeatedly encountering that model type within the CDB ***and...***
- b) the modeler is making a statement that he will likely associate the same modeled representation (same shape, size, texture, materials, or attributes, etc.) for the feature type – as a result, the client-devices can count on the fact that the model will be heavily replicated throughout the CDB. The characterization of a model as geotypical tells the consumers of the CDB that the model is heavily used throughout the CDB and that it may be cached in memory for re-use.

The manner in which geotypical models are stored / accessed differs from their geospecific counterparts. Geotypical models are stored in their own directory structure; this group of models is referred collectively as the GTModel library. The storage structure of the GTModel library provides a convenient categorization of models by their feature codes and their level-of-detail. As a result, geotypical models can be managed as a global library of 3D models that are used to fill the CDB with cultural detail.

The above discussion applies equally to statically-positioned moving models. The manner in which statically-positioned moving model features (and their modeled representations) are stored and accessed is similar to geotypical models; it differs however in the fact that the MModel library provides a categorization of models by their DIS code. The model is fetched from the MModel library regardless of whether it is used as statically-positioned model by the modeler or whether it is dynamically-positioned by the client-device during the simulation.

Conversely, when a feature is tagged as geospecific...

- a) the modeler is making a statement about his knowledge of the very low probability of encountering (typically only once) that feature type within the CDB ***or...***
- b) the modeler is making a statement regarding his intention to associate a unique modeled representation (different shape, size, texture, materials, or attributes, etc.) for that feature – as a result, the client-devices can assume that the feature will never share the same modeled representation with other features (e.g., no model replication) within the CDB. Real-world recognizable cultural point features (say the Eiffel Tower, the Pentagon, the CN Tower) are usually modeled as geospecific.

GSMODELS have a storage organization that is consistent with Tiled datasets. The storage organization of tiled datasets has been optimized to efficiently access CDB content by its lat-long location, its level-of-detail and its dataset component type.

Requirement 40

<http://opengis.net/spec/CDB/1.0/core/geospecific-storage>

Like all of the CDB Tiled datasets, geospecific models *SHALL* be stored in the \CDB\Tiles\ directory. As a result, client-devices can reference each model with a unique directory path and a unique file name which is derived from the model's unique position, level-of-detail, and its feature code

See section 6.4.7.2, Volume 10 OGC CDB Implementation Guidance for more implementation guidance on this topic.

3.3.6 Organizing Models into Levels of Detail

Requirement 41

<http://opengis.net/spec/CDB/1.0/core/lod-organization-resolution>

The geometry, texture, and signature datasets of 3D models *SHALL* be organized into levels of details (LOD) based on their resolutions.

The expression of the model resolution depends on the dataset; the resolution of the model geometry is called its Significant Size (SS); the texture resolution is expressed by its Texel Size (TS); and for the radar signature of a model, its resolution is simply its size and is measured by the diameter of its Bounding Sphere (BSD).

The lower bounds (LB) of SS, TS, and BSD for a given LOD can be expressed by the following set of equations.

$$LB_{SS} > 111319/2^{LOD+11} \text{ m}$$

$$LB_{TS} > 111319/2^{LOD+14} \text{ m}$$

$$LB_{BSD} > 111319/2^{LOD+8} \text{ m}$$

In all three equations, the value 111319 represents the approximate length in meters of an arc of one degree at the equator²².

For convenience, the following table gives the CDB LOD associated with these three measures of the resolution of a model. Note that all values are expressed in meters using a scientific notation with 6 decimals.

Table 3-1: CDB LOD vs. Model Resolution

	ModelGeometry	ModelTexture	ModelSignature
CDB LOD	Significant Size	Texel Size	Bounding Sphere
-10	SS > 5.565950 × 10 ⁺⁴	TS > 6.957438 × 10 ⁺³	BSD > 4.452760 × 10 ⁺⁵
-9	SS > 2.782975 × 10 ⁺⁴	TS > 3.478719 × 10 ⁺³	BSD > 2.226380 × 10 ⁺⁵
-8	SS > 1.391488 × 10 ⁺⁴	TS > 1.739359 × 10 ⁺³	BSD > 1.113190 × 10 ⁺⁵
-7	SS > 6.957438 × 10 ⁺³	TS > 8.696797 × 10 ⁺²	BSD > 5.565950 × 10 ⁺⁴
-6	SS > 3.478719 × 10 ⁺³	TS > 4.348398 × 10 ⁺²	BSD > 2.782975 × 10 ⁺⁴
-5	SS > 1.739359 × 10 ⁺³	TS > 2.174199 × 10 ⁺²	BSD > 1.391488 × 10 ⁺⁴
-4	SS > 8.696797 × 10 ⁺²	TS > 1.087100 × 10 ⁺²	BSD > 6.957438 × 10 ⁺³
-3	SS > 4.348398 × 10 ⁺²	TS > 5.435498 × 10 ⁺¹	BSD > 3.478719 × 10 ⁺³
-2	SS > 2.174199 × 10 ⁺²	TS > 2.717749 × 10 ⁺¹	BSD > 1.739359 × 10 ⁺³
-1	SS > 1.087100 × 10 ⁺²	TS > 1.358875 × 10 ⁺¹	BSD > 8.696797 × 10 ⁺²
0	SS > 5.435498 × 10 ⁺¹	TS > 6.794373 × 10 ⁺⁰	BSD > 4.348398 × 10 ⁺²
1	SS > 2.717749 × 10 ⁺¹	TS > 3.397186 × 10 ⁺⁰	BSD > 2.174199 × 10 ⁺²
2	SS > 1.358875 × 10 ⁺¹	TS > 1.698593 × 10 ⁺⁰	BSD > 1.087100 × 10 ⁺²
3	SS > 6.794373 × 10 ⁺⁰	TS > 8.492966 × 10 ⁻¹	BSD > 5.435498 × 10 ⁺¹
4	SS > 3.397186 × 10 ⁺⁰	TS > 4.246483 × 10 ⁻¹	BSD > 2.717749 × 10 ⁺¹

²² The actual equation to obtain the values of 111319 m is $L=a \times \text{PI}180^\circ$ where “a” is the length of the major semi-axis of the WGS-84 ellipsoid; “a” is also known as the equatorial radius.

5	SS > $1.698593 \times 10^{+0}$	TS > 2.123241×10^{-1}	BSD > $1.358875 \times 10^{+1}$
6	SS > 8.492966×10^{-1}	TS > 1.061621×10^{-1}	BSD > $6.794373 \times 10^{+0}$
7	SS > 4.246483×10^{-1}	TS > 5.308104×10^{-2}	BSD > $3.397186 \times 10^{+0}$
8	SS > 2.123241×10^{-1}	TS > 2.654052×10^{-2}	BSD > $1.698593 \times 10^{+0}$
9	SS > 1.061621×10^{-1}	TS > 1.327026×10^{-2}	BSD > 8.492966×10^{-1}
10	SS > 5.308104×10^{-2}	TS > 6.635129×10^{-3}	BSD > 4.246483×10^{-1}
11	SS > 2.654052×10^{-2}	TS > 3.317565×10^{-3}	BSD > 2.123241×10^{-1}
12	SS > 1.327026×10^{-2}	TS > 1.658782×10^{-3}	BSD > 1.061621×10^{-1}
13	SS > 6.635129×10^{-3}	TS > 8.293912×10^{-4}	BSD > 5.308104×10^{-2}
14	SS > 3.317565×10^{-3}	TS > 4.146956×10^{-4}	BSD > 2.654052×10^{-2}
15	SS > 1.658782×10^{-3}	TS > 2.073478×10^{-4}	BSD > 1.327026×10^{-2}
16	SS > 8.293912×10^{-4}	TS > 1.036739×10^{-4}	BSD > 6.635129×10^{-3}
17	SS > 4.146956×10^{-4}	TS > 5.183695×10^{-5}	BSD > 3.317565×10^{-3}
18	SS > 2.073478×10^{-4}	TS > 2.591847×10^{-5}	BSD > 1.658782×10^{-3}
19	SS > 1.036739×10^{-4}	TS > 1.295924×10^{-5}	BSD > 8.293912×10^{-4}
20	SS > 5.183695×10^{-5}	TS > 6.479619×10^{-6}	BSD > 4.146956×10^{-4}
21	SS > 2.591847×10^{-5}	TS > 3.239809×10^{-6}	BSD > 2.073478×10^{-4}
22	SS > 1.295924×10^{-5}	TS > 1.629905×10^{-6}	BSD > 1.036739×10^{-4}
23	SS > 0	TS > 0	BSD > 0

When using the table to perform a lookup, first compute the value of SS, TS, or BSD, then scan through the lines of the table starting at the top with LOD -10; when the computed value is larger than the lower bound of the LOD, select that LOD. Since the values of SS, TS, and BSD are, by definition, always positive, the search for a LOD will always be successful; in the worst case, the search will end with the last line of the table.

3.3.7 Organizing Models into Datasets

GModel, GTModel, and MModel are organized into multiple datasets representing their exterior shell and interior, and their geometry and texture. The exterior of a model is called its shell and is composed of a set of datasets representing its geometry (ModelGeometry and ModelDescriptor) and its textures (ModelTexture, ModelMaterial, and ModelCMT). Similarly, the interior of a model is divided into geometry (ModelInteriorGeometry and ModelInteriorDescriptor) and textures (ModelInteriorTexture, ModelInteriorMaterial, and ModelInteriorCMT) datasets.

The following is the requirements class for naming Models.

Requirements Class - Geotypical Models (GTModel) naming conventions (42-56)	
/req/core/gtmodel-naming	
Target type	Operations
Dependency	Various XML schema
Requirement 42	/req/core/texture-name
Requirement 43	/req/core/texture-name -file-name
Requirement 44	/req/core/lod -file-name
Requirement 45	/req/core/gtmodel-directories
Requirement 46	/req/core/gtmodelgeometry-entry-name
Requirement 47	/req/core/gtmodelgeometry-lod-name
Requirement 48	/req/core/gtmodeldescriptor-name
Requirement 49	/req/core/gtmodeltexture-name
Requirement 50	/req/core/gtmodelmaterial-name
Requirement 51	/req/core/gtmodelcmt-name
Requirement 52	/req/core/gtmodelinteriorgeometry-name
Requirement 53	/req/core/gtmodelinteriordescriptor-name
Requirement 54	/req/core/gtmodelinteriortexture-name
Requirement 55	/req/core/gtmodelinteriormaterial-name
Requirement 56	/req/core/gtmodelsignature-name

3.3.8 Terms and Expressions

When referring to 3D Models, this standard makes use of a number of terms and expressions that are frequently mentioned throughout the text; these terms and expressions are defined below.

3.3.8.1 Feature Classification

The CDB standard has an important Feature Data Dictionary (FDD) whose origins are traceable to the DIGEST v2.1 Specification. However, the current FDD is a consolidation of the DIGEST, DGIWG, SEDRIS, and UHRB dictionaries. The CDB FDD makes use of feature codes²³ to classify features. To provide an even better classification of features, the CDB standard defines an additional attribute called the feature sub-code (FSC). By extending the feature code hierarchy structure in this manner, it is possible to define a broader set of model types. The sub-code value and its significance depend on the primary feature code. Refer to [/CDB/Metadata/Feature_Data_Dictionary.xml](#) for the complete list of feature codes and subcodes.

Feature Code (aka FACC) and FSC are two CDB attributes defined in sections 5.7.1.3.24 and 5.7.1.3.25 respectively.

One of the uses of feature codes is to create a hierarchy of subdirectories by taking advantage of the manner in which a Feature Data Dictionary is built. In CDB, a feature code is a 5-character code where the first character represents a category of features, the second represents a subcategory of the current category, and the last three characters represent a specific type in the subcategory. The CDB standard uses these three parts to compose the following hierarchy of folders:

```
\A_Category\B_Subcategory\999_Type\
```

Where A is the first character of the feature code, Category is the category name, B is the second character of the feature code, Subcategory is the subcategory name, 999 are the 3rd, 4th, and 5th characters of the feature code, and Type feature type as per [/CDB/Metadata/Feature_Data_Dictionary.xml](#).

3.3.8.1.1 A note on Feature Codes

Feature codes provide a means for encoding real-world entities or objects and concepts, including those which are not necessarily visible or have a tangible physical form (e.g., airspace). Feature codes allow a standardized way to describe the world in terms of features, attributes and attribute values. Feature codes do not specify the delineation or geometry of features. Attributes are the properties or characteristics associated with features. A standardized dictionary is required to support encoding in order to maximize interoperability and to understand the production, exchange, distribution, and exploitation of digital geographic data.

²³ In CDB Version 1, Feature Codes were originally based on the FACC (see section 3.3.8.1.1). Future versions of CDB will enhance and extend the capability to allow use of other feature code vocabularies.

Feature codes are defined and stored in a dictionary of features, attributes and attribute values organized in a standardized coding system²⁴. Feature codes have not been developed to satisfy the requirements of any single application, product, or data store. Feature codes are intended to be independent from level of resolution (scale), representation, or portrayal. The appropriate selection of features codes and attributes are intended to be implemented as part of the overall solution for an application, by means of a database (supported by a data schema or model), a product, or dataset (defined according to a format specification and a data model).

Users of feature codes are advised that, as with any dictionary, there may be more than one way to encode geographic entities, either by offering a choice of features or a combination of features and attributes. A heliport is listed as feature GB035 (Heliport), but could also be encoded as feature code GB006 (Airfield) associated with the attribute APT (Airfield type) containing a coded value of 009 (Heliport). Another example would be AK090 (Fairgrounds) and AK091 (Exhibition Grounds), which could be interchanged depending on the user's own interpretation.

3.3.8.2 Model Name

When a feature is represented by a 3D model, the model itself is given a name that is used to better describe or differentiate two features having the same feature and FSC codes. Even though the model name is left to the discretion of the modeler, the CDB standard recommends the use of the feature code based name as the model name. See the `/CDB/Metadata/Feature_Data_Dictionary.xml` for the complete list of feature codes. In the case of Moving Models, the model name is the human-readable version of its DIS Entity Type.

The model name corresponds to the MODL attribute defined in section 5.7.1.3.41.

3.3.8.3 DIS Entity Type

CDB Moving Models make use of the DIS standard (see reference [7]) to create the directory structures where MModel datasets are stored. The DIS standard uses a structure called the DIS Entity Type to identify a “moving model”; this structure is made of seven fields named:

1. Kind
2. Domain
3. Country
4. Category
5. Subcategory
6. Specific
7. Extra

The first four fields (kind, domain, country and category) are used to create four subdirectories in the moving model datasets hierarchy. Each of the directory names is composed of the field's

²⁴ The CDB Feature Data Dictionary (FDD) is provided with the CDB Standard in the form of an XML file. An XML Stylesheet is provided to format and display the dictionary inside a standard Web browser. Furthermore, the XML Schema defining the format of the FDD can also be found in the Schema subdirectory of the CDB Standard Distribution Package.

value (1 to 3 digits), followed by an underscore “_”, and concatenated with the field’s name as per Annex M (Volume 2 CDB Core: Model and Physical Structure Annexes).

Another directory name is created by concatenating all fields with the underscore character. This character string also forms the Moving Model DIS Code (MMDC) attribute later defined in section 5.7.1.3.40.

Together, these five directories classify CDB Moving Models into a DIS-like structure that looks like this:

```
.\1_Kind\2_Domain\3_Country\4_Category\1_2_3_4_5_6_7\
```

The above directory structure is used, for instance, by the MModelGeometry dataset later defined in section 3.5.1.

3.3.8.4 Texture Name

Requirement 42

http://www.opengis.net/spec/cdb/1.0/core/texture-name

The name of 3D model textures <i>SHALL</i> be a character string having a minimum of 2 characters and a maximum length of 32 characters. The first two characters <i>SHALL</i> be alphanumeric.

Examples of valid texture names are Brick, M1A2, house, City_Hall, etc. A name such as C-130 is invalid because the second character (“-”) is not alphanumeric.

Requirement 43

http://www.opengis.net/spec/cdb/1.0/core/texture-name-file-name

The acronym TNAM <i>SHALL</i> represents the texture name and is used to compose texture file and directory names. The following directory structure <i>SHALL</i> be used by CDB Model texture-related datasets: \A\B\TNAM\

The directory represented by \A corresponds to the first character of TNAM in uppercase. The second directory, \B, corresponds to the second character of TNAM in uppercase. As a result, a texture named “house” will be stored in a directory tree with the following structure:

3.3.8.5 Level of Detail

The terms “Level of Detail” and its acronym “LOD” are generally well known to the intended audience of this standard.

Requirement 44

<http://www.opengis.net/spec/cdb/1.0/core/lod-file-name>

In the context of the CDB standard, filenames and directory names *SHALL* be composed from the concept of LOD.

This standard specifies a numeric scale to classify a LOD between 34 levels numbered from –10 to +23. The details will be provided later in the document. At this point, it is sufficient to define the convention used throughout the standard to designate a particular LOD.

The standard designates a LOD by appending its level to the uppercase letter L. When the level is negative, the uppercase letter C is used in lieu of the minus sign. The numeric values of all levels are represented by 2-digit numbers. As a result, LODs are designated as LC10 for level –10, L00 for level 0, or L23 for level 23.

3.4 GTModel Library Datasets

Requirement 45

<http://www.opengis.net/spec/cdb/1.0/core/gtmodel-directories>

The **\CDB\GTModel** folder *SHALL* be the root directory of the GTModel library which is composed of the following datasets:

1. GTModelGeometry
2. GTModelTexture
3. GTModelDescriptor
4. GTModelMaterial
5. GTModelCMT
6. GTModelInteriorGeometry
7. GTModelInteriorTexture
8. GTModelInteriorDescriptor
9. GTModelInteriorMaterial

- 10. GTModelInteriorCMT
- 11. GTModelSignature

These datasets are stored in five (5) different directory structures described in the subsections below.

3.4.1 GTModel Directory Structure 1: Geometry and Descriptor

This directory structure holds the geometry-related datasets of the GTModel Library; they are:

1. Dataset 500, GTModelGeometry Entry File
2. Dataset 510, GTModelGeometry Level of Detail
3. Dataset 503, GTModelDescriptor

The directory structure has 5 levels and is based on the feature code of the model (see section 3.3.8.1).

Table 3-2: GTModelGeometry Directory Structure

Director y Level	Directory Name	Description
Level 1	500_GTModelGeometr y	The name of the directory is composed of the dataset code followed by an underscore and the dataset name.
Level 2	A_Category	The first character of the feature code is called the “Feature Category”. The name of the directory is composed of the first character (denoted A) of the category name followed by an underscore and the category name (denoted Category) as per Section 1.5.
Level 3	B_Subcategory	The second character of the feature code is called the “Feature Subcategory”. The name of the directory is composed of the first character (denoted B) of the subcategory name followed by an underscore and the subcategory name (denoted Subcategory) as per Section 1.5.
Level 4	999_Type	The 3 rd , 4 th , and 5 th characters of the feature code are called the “Feature Type”. The name of the directory is composed of the feature type (denoted 999) followed by

		an underscore and the name (denoted Type) associated with the feature type as per Section 1.5.
Level 5	LOD	Character L followed by the LOD number corresponding to the Significant Size for positive levels of detail. Characters LC followed by the LOD number corresponding to the Significant Size for negative levels of detail.

3.4.1.1 GTModelGeometry Entry File Naming Convention

Requirement 46
http://www.opengis.net/spec/cdb/core/1.0/gtmodelgeometry-entry-name
The files of the GTModelGeometry Entry File dataset <i>SHALL</i> be stored in level 4 of the 5-level directory structure presented above. The names of the files <i>SHALL</i> adhere to the following naming convention: D500_Snnn_Tnnn_FeatureCode_FSC_MODL.flr

The following table defines each field of the file name and Table 5-9 provides the values of the Component Selectors to complete the name.

Table 3-3: GTModelGeometry Entry File Naming Convention

Field	Description
D500	Character D followed by the 3-digit code assigned to the dataset
Snnn	Character S followed by the 3-digit Component Selector 1
Tnnn	Character T followed by the 3-digit Component Selector 2
FeatureCode	Five-character feature code as defined in Section 1.5
FSC	Three-digit integer representing the feature sub-code (FSC) as defined in Section 1.5
MODL	32-character Model Name String
flr	The file type associated with the dataset (currently an OpenFlight file)

3.4.1.2 GTModelGeometry Level of Detail Naming Convention

Requirement 47
http://www.opengis.net/spec/cdb/core/1.0/gtmodelgeometry-lod-name
The files of the GTModelGeometry Level of Detail dataset <i>SHALL</i> be stored in level 5 of its directory structure. The names of the files <i>SHALL</i> adhere to the following naming convention: D510_Snnn_Tnnn_LOD_FeatureCode_FSC_MODL.flt

The following table defines each field of the file name and Table 5-9 provides the values of the Component Selectors to complete the name.

Table 3-4: GTModelGeometry Level of Detail Naming Convention

Field	Description
D510	Character D followed by the 3-digit code assigned to the dataset
Snnn	Character S followed by the 3-digit Component Selector 1
Tnnn	Character T followed by the 3-digit Component Selector 2
LOD	This field is identical to the name of the LOD directory (level 5) where the file is stored.
FeatureCode	Five-character feature code as defined in Section 1.5
FSC	Three-digit integer representing the feature sub-code (FSC) as defined in Section 1.5
MODL	32-character Model Name String
flt	The file type associated with the dataset (currently an OpenFlight file)

3.4.1.3 GTModelDescriptor Naming Convention

Requirement 48
http://www.opengis.net/spec/cdb/core/1.0/gtmodeldescriptor-name
The files of the GTModelDescriptor dataset <i>SHALL</i> be stored in level 4 of the 5-level directory structure presented above. The names of the files <i>SHALL</i> adhere to the

following naming convention: D503_Snnn_Tnnn_FeatureCode_FSC_MODL.xml

The following table defines each field of the file name and Table 5-9 provides the values of the Component Selectors to complete the name.

Table 3-5: GTModelDescriptor Naming Convention

Field	Description
D503	Character D followed by the 3-digit code assigned to the dataset
Snnn	Character S followed by the 3-digit Component Selector 1
Tnnn	Character T followed by the 3-digit Component Selector 2
FeatureCode	Five-character feature code as defined in Section 1.5
FSC	Three-digit integer representing the feature sub-code (FSC) as defined in Section 1.5
MODL	32-character Model Name String
xml	The file type associated with the dataset

3.4.1.4 Examples

The following example illustrates the directory structure that would store the entry file of all geotypical buildings with a feature code of AL015:

```
\CDB\GTModel\500_GTModelGeometry\A_Culture\L_Misc_Feature\  
015_Building\
```

Where \CDB\GTModel is the root of all geotypical model datasets, \500_GTModelGeometry is the directory containing all feature code categories, \A_Culture is the directory containing all feature code subcategories of category A (named Culture), \L_Misc_Feature is the directory containing all feature types of category A and subcategory L (named Misc_Feature), \015_Building is the directory containing all OpenFlight files representing geotypical buildings whose feature types are 015 (named Building).

Examples of files found in the above directory are:

```
.\D500_S001_T001_AL015_004_Castle.flt  
.\D500_S001_T001_AL015_015_School.flt  
.\D500_S001_T001_AL015_021_Garage.flt  
.\D500_S001_T001_AL015_037_Fire_Station.flt  
.\D500_S001_T001_AL015_050_Church.flt
```

Note that all filenames start with a common portion (D500_S001_T001_AL015) and that only their FSC and MODL portions vary.

If the castle above (AL015_004_Castle) is represented with 3 levels of details, say LOD 3, 5 and 8, they would be stored in .\L03\, .\L05\, and .\L08\ giving file names such as these:

```

.\L03\D510_S001_T001_L03_AL015_004_Castle.flt
.\L05\D510_S001_T001_L05_AL015_004_Castle.flt
.\L08\D510_S001_T001_L08_AL015_004_Castle.flt

```

Again, the descriptor associated with the same castle (AL015_004_Castle) would be found in this file:

```

.\D503_S001_T001_AL015_004_Castle.xml

```

3.4.2 GTModel Directory Structure 2: Texture, Material, and CMT

This directory structure holds the texture-related datasets of the GTModel Library; they are:

1. Dataset 501, GTModelTexture (Deprecated)
2. Dataset 511, GTModelTexture
3. Dataset 504, GTModelMaterial
4. Dataset 505, GTModelCMT

The directory structure has 4 levels and is based on the texture name.

Table 3-6: GTModelTexture Directory Structure

Director y Level	Directory Name	Description
Level 1	501_GTModelTexture	The name of the directory is composed of the dataset code followed by an underscore and the dataset name.
Level 2	A	The name of the directory corresponds to the first character of texture name (TNAM), in uppercase.
Level 3	B	The name of the directory corresponds to the second character of texture name (TNAM), in uppercase.
Level 4	TNAM	The texture name is between 2 and 32 characters in length. The first two

		characters are alphanumeric.
--	--	------------------------------

3.4.2.1 GTModelTexture Naming Convention

Requirement 49
http://www.opengis.net/spec/cdb/core/1.0/gtmodeltexture-name
The names of the GTModelTexture files <i>SHALL</i> adhere to the following naming convention: D511_Snnn_Tnnn_LOD_TNAM.rgb

The following table defines each field of the file name and Table 5-8 provides the values of the Component Selectors to complete the name.

Table 3-7: GTModelTexture Naming Convention

Field	Description
D511	Character D followed by the 3-digit code assigned to the dataset.
Snnn	Character S followed by the 3-digit Component Selector 1
Tnnn	Character T followed by the 3-digit Component Selector 2
LOD	The Level of Detail corresponding to the Texel Size of the texture as explained in section 3.3.8.5.
TNAM	The texture name; identical to the folder name where the texture resides.
rgb	The file type associated with the dataset (currently a SGI Image)

3.4.2.2 GTModelMaterial Naming Convention

Requirement 50
http://www.opengis.net/spec/cdb/core/1.0/gtmodelmaterial-name
The names of the GTModelMaterial files <i>SHALL</i> adhere to the following naming convention: D504_Snnn_Tnnn_LOD_TNAM.tif

The following table defines each field of the file name and Table 5-9 provides the values of the Component Selectors to complete the name.

Table 3-8: GTModelMaterial Naming Convention

Field	Description
D504	Character D followed by the 3-digit code assigned to the dataset.
Snnn	Character S followed by the 3-digit Component Selector 1
Tnnn	Character T followed by the 3-digit Component Selector 2
LOD	The Level of Detail corresponding to the Texel Size of the texture as explained in section 3.3.8.5.
TNAM	The material texture name; identical to the folder name where the material texture resides.
tif	The file type associated with the dataset (currently specified as a TIFF file)

3.4.2.3 GTModelCMT Naming Convention

Requirement 50
http://www.opengis.net/spec/cdb/core/1.0/gtmodelcmt-name
The names of the GTModelCMT files <i>SHALL</i> adhere to the following naming convention: D505_Snnn_Tnnn_TNAM.xml

The following table defines each field of the file name and Table 5-9 provides the values of the Component Selectors to complete the name.

Table 3-9: GTModelMaterial Naming Convention

Field	Description
D505	Character D followed by the 3-digit code assigned to the dataset.
Snnn	Character S followed by the 3-digit Component Selector 1
Tnnn	Character T followed by the 3-digit Component Selector 2
TNAM	The material texture name; identical to the folder name where the material texture resides.
xml	The file type associated with the dataset

3.4.2.4 Examples

The following example illustrates the directory structure that would store all files associated with a texture named ‘Brick’:

```
\CDB\GTModel\501_GTModelTexture\B\R\Brick\
```

Where \CDB\GTModel is the root of all geotypical model datasets, \501_GTModelTexture is the directory containing all geotypical textures, \B is the directory containing all textures whose name start with the letter ‘B’ or ‘b’, \R is the directory containing all textures whose name have the letter ‘R’ or ‘r’ in the second position, and \Brick is the directory containing all texture-related files whose name is ‘Brick’. Note that the second letter of the texture name is a lowercase ‘r’ but the corresponding directory name is an uppercase ‘R’.

If the Brick texture has a resolution of 1 cm and a dimension of 256 x 256 pixels, Table 3-1 tells us that the finest LOD will be 10 (TS = 0.01 m) and the coarsest will be 2 (TS = 2.56 m), and the following files would be found in the above directory:

```
.\D511_S001_T001_L02_Brick.rgb  
.\D511_S001_T001_L03_Brick.rgb  
.\D511_S001_T001_L04_Brick.rgb  
.\D511_S001_T001_L05_Brick.rgb  
.\D511_S001_T001_L06_Brick.rgb  
.\D511_S001_T001_L07_Brick.rgb  
.\D511_S001_T001_L08_Brick.rgb  
.\D511_S001_T001_L09_Brick.rgb  
.\D511_S001_T001_L10_Brick.rgb
```

The following example illustrates the directory structure that would store all LODs of a material texture whose name is Church-Gothic:

```
\CDB\GTModel\501_GTModelTexture\C\U\Church-Gothic\
```

Again, note that the second letter of the material texture name is a lowercase ‘u’ but the corresponding directory name is an uppercase ‘U’.

If the material texture has a resolution of 15 cm and a dimension of 256 x 256 pixels, the finest LOD will be 6 and the coarsest will be -2, and the following files would be found in the above directory:

```
.\D504_Snnn_Tnnn_LC02_Church-Gothic.tif  
.\D504_Snnn_Tnnn_LC01_Church-Gothic.tif  
.\D504_Snnn_Tnnn_L00_Church-Gothic.tif
```

```

.\D504_Snnn_Tnnn_L01_Church-Gothic.tif
.\D504_Snnn_Tnnn_L02_Church-Gothic.tif
.\D504_Snnn_Tnnn_L03_Church-Gothic.tif
.\D504_Snnn_Tnnn_L04_Church-Gothic.tif
.\D504_Snnn_Tnnn_L05_Church-Gothic.tif
.\D504_Snnn_Tnnn_L06_Church-Gothic.tif

```

The composite material table associated with the above material textures would reside in the same directory and be named:

```
.\D505_Snnn_Tnnn_Church-Gothic.xml
```

3.4.3 GTModel Directory Structure 3: Interior Geometry and Descriptor

This directory structure holds the datasets related to the geometry of the interior of a GTModel; they are:

1. Dataset 506, GTModelInteriorGeometry
2. Dataset 508, GTModelInteriorDescriptor

The directory structure has 5 levels and is based on the feature code of the model (see section 3.3.8.1).

Table 3-10: GTModelInteriorGeometry Directory Structure

Directory Level	Directory Name	Description
Level 1	506_GTModelInteriorGeometry	The name of the directory is composed of the dataset code followed by an underscore and the dataset name.
Level 2	A_Category	The first character of the feature code (denoted A), called the “Feature Category”, followed by an underscore and the category name (denoted Category) as per Section 1.5.
Level 3	B_Subcategory	The second character of the feature code (denoted B), called the “Feature Subcategory”, followed by an underscore and the subcategory

		name (denoted Subcategory) as per Section 1.5.
Level 4	999_Type	The 3 rd , 4 th , and 5 th characters of the feature code (denoted 999), called the “Feature Type”, followed by an underscore and the name (denoted Type) associated with the feature type as per Section 1.5.
Level 5	LOD	The Level of Detail corresponding to the Significant Size of the model as explained in section 3.3.8.5.

3.4.3.1 GTModelInteriorGeometry Naming Convention

Requirement 52
http://www.opengis.net/spec/cdb/core/1.0/gtmodelinteriorgeometry-name
The files of the GTModelInteriorGeometry dataset <i>SHALL</i> be stored in level 5 of its directory structure. The names of the files <i>SHALL</i> adhere to the following naming convention: D506_Snnn_Tnnn_LOD_FeatureCode_FSC_MODL.flt

The following table defines each field of the file name and Table 5-9 provides the values of the Component Selectors to complete the name.

Table 3-11: GTModelInteriorGeometry Naming Convention

Field	Description
D506	Character D followed by the 3-digit code assigned to the dataset
Snnn	Character S followed by the 3-digit Component Selector 1
Tnnn	Character T followed by the 3-digit Component Selector 2
LOD	This field is identical to the name of the LOD directory (level 5) where the file is stored.
FeatureCode	Five-character feature code as defined in Section 1.5
FSC	Three-digit integer representing the feature sub-code (FSC) as defined in Section 1.5
MODL	32-character Model Name String

flt	The file type associated with the dataset (Currently specified as an OpenFlight file)
-----	---

3.4.3.2 GTModelInteriorDescriptor Naming Convention

Requirement 53
http://www.opengis.net/spec/cdb/core/1.0/gtmodelinteriordescriptor-name
The files of the GTModelInteriorDescriptor dataset <i>SHALL</i> be stored in level 4 of the 5-level directory structure presented above. The names of the files <i>SHALL</i> adhere to the following naming convention: D508_Snnn_Tnnn_FC_FSC_MODL.xml

The following table defines each field of the file name and Table 5-9 provides the values of the Component Selectors to complete the name.

Table 3-12: GTModelInteriorDescriptor Naming Convention

Field	Description
D508	Character D followed by the 3-digit code assigned to the dataset
Snnn	Character S followed by the 3-digit Component Selector 1
Tnnn	Character T followed by the 3-digit Component Selector 2
FC	(Feature Code) Five-character feature code as defined in Section 1.5
FSC	Three-digit integer representing the feature sub-code (FSC) as defined in Section 1.5
MODL	32-character Model Name String
xml	The file type associated with the dataset.

3.4.3.3 Examples

The following example illustrates the directory structure that would store the interior of all geotypical buildings represented at LOD 3 and whose feature code is AL015:

```
\CDB\GTModel\506_GTModelInteriorGeometry\A_Culture\  
L_Misc_Feature\015_Building\LO3\
```

Where \CDB\GTModel is the root of all geotypical model datasets, \505_GTModelInteriorGeometry is the directory containing all feature categories, \A_Culture is the directory containing all feature subcategories of category A (named Culture), \L_Misc_Feature is the directory containing all feature types of category A and subcategory L

(named Misc_Feature), \015_Building is the directory containing all level of details representing geotypical buildings whose feature types are 015 (named Building), and \L03 is the directory containing the model files²⁵ representing LOD 3 of these buildings.

Examples of files found in the above directory are:

```
.\D506_S001_T001_L03_AL015_004_Castle.flt  
.\D506_S001_T001_L03_AL015_015_School.flt  
.\D506_S001_T001_L03_AL015_021_Garage.flt  
.\D506_S001_T001_L03_AL015_037_Fire_Station.flt  
.\D506_S001_T001_L03_AL015_050_Church.flt
```

Note that all filenames start with a common portion (D506_S001_T001_L03_AL015) and that only their FSC and MODL portions vary.

The descriptors associated with the interior of these models would be found in level 4 of the directory structure in the following files:

```
\CDB\GTModel\506_GTModelInteriorGeometry\A_Culture\  
L_Misc_Feature\015_Building\  
.\D508_S001_T001_AL015_004_Castle.xml  
.\D508_S001_T001_AL015_015_School.xml  
.\D508_S001_T001_AL015_021_Garage.xml  
.\D508_S001_T001_AL015_037_Fire_Station.xml  
.\D508_S001_T001_AL015_050_Church.xml
```

3.4.4 GTModel Directory Structure 4: Interior Texture, Material, and CMT

This directory structure holds the datasets related to the textures of the interior of a GTModel; they are:

1. Dataset 507, GTModelInteriorTexture
2. Dataset 509, GTModelInteriorMaterial
3. Dataset 513, GTModelInteriorCMT

The directory structure has 4 levels and is based on the texture name.

²⁵ Currently OpenFlight

Table 3-13: GTModelInteriorTexture Directory Structure

Director y Level	Directory Name	Description
Level 1	507_GTModelInteriorTexture	The name of the directory is composed of the dataset code followed by an underscore and the dataset name.
Level 2	A	The name of the directory corresponds to the first character of texture name (TNAM), in uppercase.
Level 3	B	The name of the directory corresponds to the second character of texture name (TNAM), in uppercase.
Level 4	TNAM	The texture name has from 2 to 32 characters. The first two characters are alphanumeric.

3.4.4.1 GTModelInteriorTexture Naming Convention

Requirement 54
http://www.opengis.net/spec/cdb/core/1.0/gtmodelinteriortexture-name
The names of the GTModelInteriorTexture files <i>SHALL</i> adhere to the following naming convention: D507_Snnn_Tnnn_LOD_TNAM.rgb

The following table defines each field of the file name and Table 5-8 provides the values of the Component Selectors to complete the name.

Table 3-14: GTModelInteriorTexture Naming Convention

Field	Description
D507	Character D followed by the 3-digit code assigned to the dataset
Snnn	Character S followed by the 3-digit Component Selector 1

Tnnn	Character T followed by the 3-digit Component Selector 2
LOD	The Level of Detail corresponding to the Texel Size of the texture as explained in section 3.3.8.5.
TNAM	The texture name; identical to the folder name where the texture resides.
rgb	The file type associated with the dataset (SGI Image)

3.4.4.2 GTModelInteriorMaterial Naming Convention

Requirement 55
http://www.opengis.net/spec/cdb/core/1.0/gtmodelinteriormaterial-name
The names of the GTModelInteriorMaterial files <i>SHALL</i> adhere to the following naming convention: D509_Snnn_Tnnn_LOD_TNAM.tif

The following table defines each field of the file name and Table 5-9 provides the values of the Component Selectors to complete the name.

Table 3-15: GTModelInteriorMaterial Naming Convention

Field	Description
D509	Character D followed by the 3-digit code assigned to the dataset
Snnn	Character S followed by the 3-digit Component Selector 1
Tnnn	Character T followed by the 3-digit Component Selector 2
LOD	The Level of Detail corresponding to the Texel Size of the texture as explained in section 3.3.8.5.
TNAM	The material texture name; identical to the folder name where the material texture resides.
tif	The file type associated with the dataset (currently specified as a TIFF file)

3.4.4.3 Example 1

The following example illustrates the directory structure that would store all LODs of a texture whose name is BeigeGypseWall:

```
\CDB\GTModel\507_GTModelInteriorTexture\B\E\BeigeGypseWall\
```

Where \CDB\GTModel is the root of all geotypical model datasets, \507_GTModelInteriorTexture is the directory containing all geotypical interior textures, \B is the directory containing all textures whose name start with the letter B, \R is the directory

containing all textures whose name start the letter ‘B’ followed by the letter ‘E’, and \BeigeGypseWall is the directory containing all LODs of the texture representing a beige gypse wall. Note that the second letter of the texture name is a lowercase ‘e’ but the corresponding directory name is an uppercase ‘E’.

If the texture has a resolution of 1 cm and a dimension of 256 x 256 pixels, the finest LOD will be 10 and the coarsest will be 2, and the following files would be found in the above directory:

```
D507_S001_T001_L02_BeigeGypseWall.rgb  
D507_S001_T001_L03_BeigeGypseWall.rgb  
D507_S001_T001_L04_BeigeGypseWall.rgb  
D507_S001_T001_L05_BeigeGypseWall.rgb  
D507_S001_T001_L06_BeigeGypseWall.rgb  
D507_S001_T001_L07_BeigeGypseWall.rgb  
D507_S001_T001_L08_BeigeGypseWall.rgb  
D507_S001_T001_L09_BeigeGypseWall.rgb  
D507_S001_T001_L10_BeigeGypseWall.rgb
```

3.4.4.4 Example 2

The following example illustrates the directory structure that would store all LODs of a material texture associated with the interior of a gothic church and whose name is Church-Gothic:

```
\CDB\GTModel\507_GTModelInteriorTexture\C\H\Church-Gothic\
```

Where \CDB\GTModel is the root of all geotypical model datasets, \507_GTModelInteriorTexture is the directory containing all geotypical interior material textures, \C is the directory containing all textures whose name start with the letter C, \H is the directory containing all textures whose name start the letter ‘C’ followed by the letter ‘H’, and \Church-Gothic is the directory containing all LODs of the material texture called Church-Gothic. Note that the second letter of the material texture name is a lowercase ‘h’ but the corresponding directory name is an uppercase ‘H’.

If the material texture has a resolution of 1 cm and a dimension of 256 x 256 pixels, the finest LOD will be 10 and the coarsest will be 2, and the following files would be found in the above directory:

```
D509_Snnn_Tnnn_L02_Church-Gothic.tif  
D509_Snnn_Tnnn_L03_Church-Gothic.tif  
D509_Snnn_Tnnn_L04_Church-Gothic.tif  
D509_Snnn_Tnnn_L05_Church-Gothic.tif  
D509_Snnn_Tnnn_L06_Church-Gothic.tif
```

D509_Snnn_Tnnn_L07_Church-Gothic.tif
D509_Snnn_Tnnn_L08_Church-Gothic.tif
D509_Snnn_Tnnn_L09_Church-Gothic.tif
D509_Snnn_Tnnn_L10_Church-Gothic.tif

3.4.5 GTModel Directory Structure 5: Signature

This directory structure holds the datasets related to the radar signature of a GTModel; they are:

1. Dataset 502, GTModelSignature (Deprecated)
2. Dataset 512, GTModelSignature

The directory structure has 5 levels and is based on the feature code of the model (see section 3.3.8.1).

Table 3-16: GTModelSignature Directory Structure

Director y Level	Directory Name	Description
Level 1	502_GTModelSignature	The name of the directory is composed of the dataset code followed by an underscore and the dataset name.
Level 2	A_Category	The first character of the feature code is called the “Feature Category”. The name of the directory is composed of the first character (denoted A) of the category name followed by an underscore and the category name (denoted Category) as per Section 1.5.
Level 3	B_Subcategory	The second character of the feature code is called the “Feature Subcategory”. The name of the directory is composed of the first character (denoted B) of the subcategory name followed by an underscore and the subcategory name (denoted Subcategory) as per Section 1.5.
Level 4	999_Type	The 3 rd , 4 th , and 5 th characters of the feature code are called the “Feature Type”. The name of the directory is composed of

		the feature type (denoted 999) followed by an underscore and the name (denoted Type) associated with the feature type as per Section 1.5.
Level 5	LOD	Character L followed by the LOD number corresponding to the Significant Size for positive levels of detail. Characters LC followed by the LOD number corresponding to the Significant Size for negative levels of detail.

Note that for compatibility with version 3.0 of the standard, the name of the directory at level 1 is kept to 502_GTModelSignature even though dataset 502 has been deprecated and replaced with dataset 512 in version 3.1 of the standard.

3.4.5.1 GTModelSignature Naming Convention

Requirement 56
http://www.opengis.net/spec/cdb/core/1.0/gtmodelsignature-name
The names of the GTModelSignature files <i>SHALL</i> adhere to the following naming convention: D512_Snnn_Tnnn_LOD_FC_FSC_MODL.xxx

The following table defines each field of the file name and Table 5-9 provides the values of the Component Selectors to complete the name.

Table 3-17: GTModelSignature Naming Convention

Field	Description
D512	Character D followed by the 3-digit code assigned to the dataset
Snnn	Character S followed by the 3-digit Component Selector 1
Tnnn	Character T followed by the 3-digit Component Selector 2
LOD	This field is identical to the name of the LOD directory (level 5) where the file is stored.
FC	(Feature Code) Five-character feature code as defined in Section 1.5
FSC	Three-digit integer representing the feature sub-code (FSC) as defined in

	Section 1.5
MODL	32-character Model Name String
xxx	The file type associated with the dataset. This is designated by the file extension; e.g. .tif for TIFF, .flt for OpenFlight, and so on.

3.4.5.2 Examples

The following example illustrates the directory structure that would store the signature of all geotypical buildings represented at LOD 3 and whose feature code is AL015:

```
\CDB\GTModel\502_GTModelSignature\A_Culture\L_Misc_Feature\
  015_Building\L03\
```

Where \CDB\GTModel is the root of all geotypical model datasets, \502_GTModelSignature is the directory containing all feature categories, \A_Culture is the directory containing all feature subcategories of category A (named Culture), \L_Misc_Feature is the directory containing all feature types of category A and subcategory L (named Misc_Feature), \015_Building is the directory containing all level of details representing the signature of geotypical buildings whose feature types are 015 (named Building), and \L03 is the directory containing the vector data sets representing LOD 3 of these buildings²⁶.

Examples of files that could be found in the above directory are:

```
.\D512_S001_T001_L03_AL015_004_Castle.shp
.\D512_S001_T001_L03_AL015_004_Castle.shx
.\D512_S001_T001_L03_AL015_004_Castle.dbf
.\D512_S001_T017_L03_AL015_004_Castle.dbf
```

3.4.6 GTModel Complete Examples

The following examples illustrate the locations and names of all files of the GTModel Library.

```
\CDB\GTModel\500_GTModelGeometry\A_Culture\L_Misc_Feature\
  015_Building\
  D500_S001_T001_AL015_004_Castle.flt           (Entry File)
  D503_S001_T001_AL015_004_Castle.xml         (Descriptor)
  Lnn\D510_S001_T001_Lnn_AL015_004_Castle.flt (LOD)

\CDB\GTModel\501_GTModelTexture\C\A\Castle\
  D511_Snnn_Tnnn_Lnn_Castle.rgb              (Texture)
  D504_Snnn_Tnnn_Lnn_Castle.tif              (Material)
  D505_Snnn_Tnnn_Castle.xml                  (CMT)
```

²⁶ Currently encoded as ShapeFiles but additional formats will be defined in a future CDB version.

```

\CDB\GTModel\502_GTModelSignature\A_Culture\L_Misc_Feature\
  015_Building\Lnn\
  D512_Snnn_Tnnn_Lnn_AL015_004_Castle.shp           (Signature)
  D512_Snnn_Tnnn_Lnn_AL015_004_Castle.shx
  D512_Snnn_Tnnn_Lnn_AL015_004_Castle.dbf
  D512_Snnn_Tnnn_Lnn_AL015_004_Castle.dbt

\CDB\GTModel\506_GTModelInteriorGeometry\A_Culture\
  L_Misc_Feature\015_Building\
  D508_S001_T001_AL015_004_Castle.xml               (Descriptor)
  Lnn\D506_S001_T001_Lnn_AL015_004_Castle.flt      (LOD)

\CDB\GTModel\507_GTModelInteriorTexture\C\A\Castle\
  D507_Snnn_Tnnn_Lnn_Castle.rgb                     (Texture)
  D509_Snnn_Tnnn_Lnn_Castle.tif                     (Material)
  D513_Snnn_Tnnn_Castle.xml                         (CMT)

```

3.5 MModel Library Datasets

The following is the requirements class for MModel Library datasets.

Requirements Class - MModel) naming conventions (57-63)	
/req/core/mmodel-naming	
Target type	Operations
Dependency	Various XML schema
Requirement 57	/req/core/mmodel-root
Requirement 58	/req/core/mmodelgeometry-name
Requirement 59	/req/core/mmodeldescriptor-name
Requirement 60	/req/core/mmodeltexture-name
Requirement 61	/req/core/mmodelmaterial-name
Requirement 62	/req/core/mmodelcmt-name
Requirement 63	/req/core/mmodelsignature-name

Requirement 57

<http://www.opengis.net/spec/cdb/core/1.0/mmodel-root>

The \CDB\MModel\ folder *SHALL* be the root directory of the MModel library which *SHALL* be composed of the following datasets.

1. MModelGeometry
2. MModelDescriptor
3. MModelTexture
4. MModelMaterial
5. MModelCMT
6. MModelSignature

These datasets are stored in three (3) different directory structures described in the subsections below.

3.5.1 MModel Directory Structure 1: Geometry and Descriptor

This directory structure is owned by the MModelGeometry dataset that is assigned dataset code 600. The structure has 6 levels and is based on the DIS Entity Type (see section 3.3.8.3). The same directory structure is used to store the files of the MModelDescriptor dataset.

Table 3-18: MModelGeometry Directory Structure

Director y Level	Directory Name	Description
Level 1	600_MModelGeometry	The name of the directory is composed of the dataset code followed by an underscore and the dataset name.
Level 2	9_Kind	The numeric code assigned to the DIS Entity Kind followed by an underscore and the name of this kind as per Annex M ²⁷ .
Level 3	9_Domain	The numeric code assigned to the DIS Domain followed by an underscore and the name of the domain as per Annex M.
Level 4	9_Country	The numeric code assigned to the DIS Country followed by an underscore and

²⁷ Volume 2 CDB Core Model and Physical Structure: Informative Annexes

		the name of this country as per Annex M.
Level 5	9_Category	The numeric code assigned to the DIS Category followed by an underscore and the name of this category as per Annex M.
Level 6	9_9_9_9_9_9_9	All 7 fields of the DIS Entity type concatenated and separated by an underscore.

3.5.1.1 MModelGeometry Naming Convention

Requirement 58
http://www.opengis.net/spec/cdb/core/1.0/mmodelgeometry-name
The names of all MModelGeometry files <i>SHALL</i> adhere to the following naming convention: D600_Snnn_Tnnn_MMDC.flr

The following table defines each field of the file names and Table 5-10 provides the values of the Component Selectors to complete the name.

Table 3-19: MModelGeometry Naming Convention

Field	Description
D600	Character D followed by the 3-digit code assigned to the dataset.
Snnn	Character S followed by the 3-digit value of Component Selector 1
Tnnn	Character T followed by the 3-digit value of Component Selector 2
MMDC	The Moving Model DIS Code is the same as directory level 6 above
flr	The file type associated with the dataset (Currently defined as an OpenFlight file)

3.5.1.2 MModelDescriptor Naming Convention

Requirement 59
http://www.opengis.net/spec/cdb/core/1.0/mmodeldescriptor-name

The MModelDescriptor dataset *SHALL* be assigned dataset code 603 and the names of all MModelDescriptor files adhere to the following naming convention:
D603_S001_T001_MMDC.xml

The following table defines each field of the file names and Table 5-10 provides the values of the Component Selectors to complete the name.

Table 3-20: MModelDescriptor Naming Convention

Field	Description
D603	Character D followed by the 3-digit code assigned to the dataset.
S001	Character S followed by the 3-digit value of Component Selector 1
T001	Character T followed by the 3-digit value of Component Selector 2
MMDC	The Moving Model DIS Code is the same as directory level 6 above
xml	The file type associated with the dataset (XML File)

3.5.1.3 Examples

The following example illustrates the directory structure that would store the M1A2 SEP version of the M1 Abrams tank.

```
\CDB\MModel\600_MModelGeometry\1_Platform\1_Land\
  225_United_States\1_Tank\1_1_225_1_1_8_0\
```

Where \CDB\MModel is the root of all moving model datasets, \600_MModelGeometry is the directory containing the geometry and descriptor of all moving models, \1_Platform is the directory containing all DIS Entity of Kind 1 (named Platform), \1_Land is the directory containing all DIS platforms of Domain 1 (named Land), \225_United_States is the directory containing all DIS land platforms of Country 225 (called United_States), \1_Tank is the directory containing all DIS land platforms of Category 1 (named Tank), and \1_1_225_1_1_8_0 is the directory containing all geometry and descriptor files of the M1A2 SEP Abrams tank.

Examples of files found in the above directory are:

```
D600_S001_T001_1_1_225_1_1_8_0.flr
D603_S001_T001_1_1_225_1_1_8_0.xml
```

3.5.2 MModel Directory Structure 2: Texture, Material, and CMT

This directory structure is owned by the MModelTexture dataset that is assigned dataset code 601. The structure has 4 levels and is based on the texture name (see section 3.3.8.4). The same directory structure is used to store the files of the MModelMaterial and MModelCMT datasets.

Table 3-21: MModelTexture Directory Structure

Director y Level	Directory Name	Description
Level 1	601_MModelTexture	The name of the directory is composed of the dataset code followed by an underscore and the dataset name.
Level 2	A	The name of the directory corresponds to the first character of the texture name (TNAM), in uppercase.
Level 3	B	The name of the directory corresponds to the second character of the texture name (TNAM), in uppercase.
Level 4	TNAM	The texture name is from 2 to 32 characters in length. The first two characters are alphanumeric.

3.5.2.1 MModelTexture Naming Convention

Requirement 60
http://www.opengis.net/spec/cdb/core/1.0/mmodeltexture-name
The names of all MModelTexture files <i>SHALL</i> adhere to the following naming convention: D601_Snnn_Tnnn_Wnn_TNAM.rgb

The following table defines each field of the file names and Table 5-8 provides the values of the Component Selectors to complete the name.

Table 3-22: MModelTexture Naming Convention

Field	Description
D601	Character D followed by the 3-digit code assigned to the dataset.
Snnn	Character S followed by the 3-digit value of Component Selector 1

Tnnn	Character T followed by the 3-digit value of Component Selector 2
Wnn	Character W followed by the 2-digit Texture Size Code
TNAM	The texture name; identical to directory level 4 above
rgb	The file type associated with the dataset (Currently defined as a SGI Image)

3.5.2.2 MModelMaterial Naming Convention

Requirement 61
http://www.opengis.net/spec/cdb/core/1.0/mmodelmaterial-name
The MModelMaterial dataset is assigned dataset code 604 and the names of all MModelMaterial files <i>SHALL</i> adhere to the following naming convention: D604_Snnn_Tnnn_Wnn_TNAM.tif

The following table defines each field of the file names and Table 5-10 provides the values of the Component Selectors to complete the name.

Table 3-23: MModelMaterial Naming Convention

Field	Description
D604	Character D followed by the 3-digit code assigned to the dataset.
Snnn	Character S followed by the 3-digit value of Component Selector 1
Tnnn	Character T followed by the 3-digit value of Component Selector 2
Wnn	Character W followed by the 2-digit Texture Size Code
TNAM	The texture name; identical to directory level 4 above
tif	The file type associated with the dataset (Currently defined as a TIFF File)

3.5.2.3 MModelCMT Naming Convention

Requirement 62
http://www.opengis.net/spec/cdb/core/1.0/mmodelcmt-name

The MModelCMT dataset is assigned dataset code 605 and the names of all MModelCMT files *SHALL* adhere to the following naming convention:
D605_S001_T001_TNAM.xml

The following table defines each field of the file names and Table 5-10 provides the values of the Component Selectors to complete the name.

Table 3-24: MModelMaterial Naming Convention

Field	Description
D605	Character D followed by the 3-digit code assigned to the dataset
S001	Character S followed by the 3-digit value of Component Selector 1
T001	Character T followed by the 3-digit value of Component Selector 2
TNAM	The texture name; identical to directory level 4 above
xml	The file type associated with the dataset

3.5.2.4 Examples

Assuming that the textures, materials, and CMT of the M1A2 SEP are called M1A2_SEP, the following directory structure would store them.

```
\CDB\MModel\601_MModelTexture\M\1\M1A2_SEP\
```

Where \CDB\MModel is the root of all moving model datasets, \601_MModelTexture is the directory containing the textures, material textures, and CMTs of all moving models, \M is the directory containing all files whose TNAM field starts with the letter 'm' or 'M', \1 is the directory containing all files whose TNAM field starts with 'm1' or 'M1', and \M1A2_SEP is the directory containing all texture-related files whose TNAM is M1A2_SEP.

Examples of files found in the above directory are:

```
D601_S005_T001_W10_M1A2_SEP.rgb
D604_S001_T001_W09_M1A2_SEP.tif
D605_S001_T001_M1A2_SEP.xml
```

3.5.3 MModel Directory Structure 3: Signature

This directory structure is dedicated to the MModelSignature dataset that is assigned dataset code 606. The structure has 7 levels and is based on the DIS Entity Type (see section 3.3.8.3).

Table 3-25: MModelSignature Directory Structure

Director y Level	Directory Name	Description
Level 1	606_MModelSignature	The name of the directory is composed of the dataset code followed by an underscore and the dataset name.
Level 2	9_Kind	The numeric code assigned to the DIS Entity Kind followed by an underscore and the name of this kind as per Annex M ²⁸ .
Level 3	9_Domain	The numeric code assigned to the DIS Domain followed by an underscore and the name of the domain as per Annex M.
Level 4	9_Country	The numeric code assigned to the DIS Country followed by an underscore and the name of this country as per Annex M.
Level 5	9_Category	The numeric code assigned to the DIS Category followed by an underscore and the name of this category as per Annex M.
Level 6	9_9_9_9_9_9_9	All 7 fields of the DIS Entity type concatenated and separated by an underscore.
Level 7	LOD	Character L followed by the LOD number corresponding to the Significant Size for positive levels of detail. Characters LC followed by the LOD number corresponding to the Significant Size for negative levels of detail.

3.5.3.1 Naming Convention

Requirement 63

²⁸ Volume 2 CDB Core: Model and Physical Structure: Informative Annexes

<http://www.opengis.net/spec/cdb/core/1.0/mmodelsignature-name>

The names of all MModelSignature files *SHALL* adhere to the following naming convention:

- D606_Snnn_Tnnn_LOD_MMDC.xxx

where ‘xxx’ is any necessary extension to uniquely identify the table or file. In the case in which multiple tables or files are necessary to store the model signature content, then the same base name *SHALL* be used and the necessary name extensions applied as required by the database or file storage technology.

The following table defines each field of the file names and Table 5-10 provides the values of the Component Selectors to complete the name.

Table 3-26: MModelSignature Naming Convention

Field	Description
D606	Character D followed by the 3-digit code assigned to the dataset.
Snnn	Character S followed by the 3-digit value of Component Selector 1
Tnnn	Character T followed by the 3-digit value of Component Selector 2
LOD	This field is identical to the name of the LOD directory (level 7) where the file is stored.
MMDC	The Moving Model DIS Code is the same as directory level 6
xxx	‘xxx’ is any necessary extension to uniquely identify the table or file. In the case in which multiple tables or files are necessary to store the model signature content, then the same base name <i>SHALL</i> be used and the necessary name extensions applied as required by the database or file storage technology.

3.5.3.2 Examples

The following example illustrates the directory structure that would store LOD 4 of the RCS Signature of the M1A2 SEP Abrams tank.

```
\CDB\MModel\606_MModelSignature\1_Platform\1_Land\  
225_United_States\1_Tank\1_1_225_1_1_8_0\L04
```

Where \CDB\MModel is the root of all moving model datasets, \606_MModelGeometry is the directory containing the RCS signature of all moving models, \1_Platform is the directory

containing all DIS Entity of Kind 1 (named Platform), \1_Land is the directory containing all DIS platforms of Domain 1 (named Land), \225_United_States is the directory containing all DIS land platforms of Country 225 (called United_States), \1_Tank is the directory containing all DIS land platforms of Category 1 (named Tank), \1_1_225_1_1_8_0 is the directory containing all levels of detail of the RCS signature of the M1A2 SEP Abrams tank, and \L04 is the directory containing the files representing LOD 4 of RCS signature of the tank.

For ShapeFiles an example of files found in the above directory are:

```
D606_Sxxx_Txxx_L04_1_1_225_1_1_8_0.shp
D606_Sxxx_Txxx_L04_1_1_225_1_1_8_0.shx
D606_Sxxx_Txxx_L04_1_1_225_1_1_8_0.dbf
D606_Sxxx_Txxx_L04_1_1_225_1_1_8_0.dbt
```

3.5.4 MModel Complete Examples

The following examples, based on the M1A2 SEP, illustrate the naming conventions of all MModel datasets.

```
\CDB\MModel\600_MModelGeometry\1_Platform\1_Land
  \225_United_States\1_Tank\1_1_225_1_1_8_0\
    D600_Snnn_Tnnn_1_1_225_1_1_8_0.flt           (Geometry)
    D603_S001_T001_1_1_225_1_1_8_0.xml         (Descriptor)
```

```
\CDB\MModel\601_MModelTexture\M\1\M1A2_SEP\
  D601_Snnn_Tnnn_Wnn_M1A2_SEP.rgb             (Texture)
  D604_Snnn_Tnnn_Wnn_M1A2_SEP.tif             (Material)
  D605_S001_T001_M1A2_SEP.xml                 (CMT)
```

```
\CDB\MModel\606_MModelSignature\1_Platform\1_Land
  \225_United_States\1_Tank\1_1_225_1_1_8_0\Lnn\
    D606_Snnn_Tnnn_Lnn_1_1_225_1_1_8_0.shp     (Signature)
    D606_Snnn_Tnnn_Lnn_1_1_225_1_1_8_0.shx
    D606_Snnn_Tnnn_Lnn_1_1_225_1_1_8_0.dbf
    D606_Snnn_Tnnn_Lnn_1_1_225_1_1_8_0.dbt
```

3.6 CDB Tiled Datasets

The \CDB\Tiles\ folder is the root directory of all tiled datasets. They all share a similar directory structure described below. All tiled datasets implement the CDB tiling scheme described in Section 2.1, Partitioning the Earth into Tiles.

Requirements Class - Tiled Datasets (64-67)

/req/core/tiled-data	
Target type	Operations
Dependency	Various XML schema
Requirement 64	/req/core/vector-dataset-limit
Requirement 65	/req/core/latitude-directory-name
Requirement 66	/req/core/longitude-directory-name
Requirement 67	/req/core/uref-directory-name

3.6.1 Tiled Dataset Types

There are three principal types of tiled datasets:

1. Raster Datasets
2. Vector Datasets
3. Model Datasets

3.6.1.1 Raster Datasets

Data elements within a tile are organized into a regular grid where data elements are evenly positioned at every $XUnit_{LOD}$ and $YUnit_{LOD}$ as described in Section 2.1.2, Tile Levels-of-Detail (Tile-LODs). This type of organization is referred to as a Raster Dataset. Raster Datasets always have a fixed number of elements corresponding to the number of units shown in Table 2-4: CDB LOD versus Tile and Grid Size. An example of a raster dataset is terrain imagery.

Note: Partially-filled Tile-LODs are not permitted in a compliant the CDB data store. In the case where data at the Tile-LOD's resolution does not fully cover the Tile-LOD's geographic footprint, the modeler (or the tools) should fill the remainder area of the Tile-LOD with the "best available" data. There are two cases to consider:

Case I: In the case where coarser LODm data exists for the remainder area of the Tile-LODn, the LODm data should be interpolated to LODn.

Case II: In the case where coarser LODm does not exist for the remainder area of the Tile-LODn, then the remainder area of Tile-LODn should be filled with the default value for this dataset.

3.6.1.2 Vector Datasets

The point features, the lineal features, and the polygon features of the CDB are organized into several Vector Datasets and into levels of details.

The level-of-detail organization of the Vector Datasets mimics the concept of map scaling commonly found in cartography (for example a 1:50,000 map). If we pursue the analogy with

cartography, increasing the LOD number (increasingly finer detail) of a dataset is equivalent to decreasing the map's scaling (**1:n** map scaling where **n** is decreasing). As is the case with cartography, the Tile-LOD number provides a clear indication of both the positional accuracy and of the density of features. Consequently, the CDB specifies an average value for the density of features for each LOD of the Vector Dataset hierarchy. Table 3-27 below defines these values. For each CDB LOD, the table provides the maximum number of points allowed per Tile-LOD and the resulting average Feature Density.

Table 3-27: CDB LOD versus Feature Density

CDB LOD	Maximum Number of Points per Tile	Approximate Tile Edge Size (meters)	Average Point Density (points/m ²)
-10	1	$1.11319 \times 10^{+05}$	8.06977×10^{-11}
-9	1	$1.11319 \times 10^{+05}$	8.06977×10^{-11}
-8	1	$1.11319 \times 10^{+05}$	8.06977×10^{-11}
-7	1	$1.11319 \times 10^{+05}$	8.06977×10^{-11}
-6	4	$1.11319 \times 10^{+05}$	3.22791×10^{-10}
-5	16	$1.11319 \times 10^{+05}$	1.29116×10^{-09}
-4	64	$1.11319 \times 10^{+05}$	5.16466×10^{-09}
-3	256	$1.11319 \times 10^{+05}$	2.06586×10^{-08}
-2	1024	$1.11319 \times 10^{+05}$	8.26345×10^{-08}
-1	4096	$1.11319 \times 10^{+05}$	3.30538×10^{-07}
0	16384	$1.11319 \times 10^{+05}$	1.32215×10^{-06}
1	16384	$5.56595 \times 10^{+04}$	5.28861×10^{-06}
2	16384	$2.78298 \times 10^{+04}$	2.11544×10^{-05}
3	16384	$1.39149 \times 10^{+04}$	8.46177×10^{-05}
4	16384	$6.95744 \times 10^{+03}$	3.38471×10^{-04}
5	16384	$3.47872 \times 10^{+03}$	1.35388×10^{-03}
6	16384	$1.73936 \times 10^{+03}$	5.41553×10^{-03}
7	16384	$8.69680 \times 10^{+02}$	2.16621×10^{-02}
8	16384	$4.34840 \times 10^{+02}$	8.66485×10^{-02}
9	16384	$2.17420 \times 10^{+02}$	3.46594×10^{-01}
10	16384	$1.08710 \times 10^{+02}$	$1.38638 \times 10^{+00}$
11	16384	$5.43550 \times 10^{+01}$	$5.54551 \times 10^{+00}$
12	16384	$2.71775 \times 10^{+01}$	$2.21820 \times 10^{+01}$
13	16384	$1.35887 \times 10^{+01}$	$8.87281 \times 10^{+01}$
14	16384	$6.79437 \times 10^{+00}$	$3.54912 \times 10^{+02}$
15	16384	$3.39719 \times 10^{+00}$	$1.41965 \times 10^{+03}$
16	16384	$1.69859 \times 10^{+00}$	$5.67860 \times 10^{+03}$
17	16384	8.49297×10^{-01}	$2.27144 \times 10^{+04}$
18	16384	4.24648×10^{-01}	$9.08576 \times 10^{+04}$
19	16384	2.12324×10^{-01}	$3.63430 \times 10^{+05}$
20	16384	1.06162×10^{-01}	$1.45372 \times 10^{+06}$
21	16384	5.30810×10^{-02}	$5.81489 \times 10^{+06}$
22	16384	2.65405×10^{-02}	$2.32595 \times 10^{+07}$

23	16384	1.32703×10^{-02}	$9.30382 \times 10^{+07}$
----	-------	---------------------------	---------------------------

Requirement 64

<http://www.opengis.net/spec/cdb/1.0/core/vector-dataset-limit>

For positive LODs, each Tile-LOD of the vector datasets *SHALL* have no more than 16,384 points to describe the features, whether the file contains point, lineal, or polygon features. For negative LODs, this limit *SHALL* be recursively divided by 4 until it reaches the value 1.

3.6.1.3 Model Datasets

The last type of tiled datasets is used to store 2D and 3D Models and will be later described in their own sections.

3.6.2 Tiled Dataset Directory Structure

The vast majority of CDB datasets are tiled; the complete list follows.

1. Elevation
2. MinMaxElevation
3. MaxCulture
4. Imagery
5. RMTTexture
6. RMDDescriptor
7. GSFeature
8. GTFeature
9. GeoPolitical
10. VectorMaterial
11. RoadNetwork
12. RailRoadNetwork
13. PowerLineNetwork
14. HydrographyNetwork
15. GSModelGeometry
16. GSModelTexture
17. GSModelSignature
18. GSModelDescriptor
19. GSModelMaterial
20. GSModelInteriorGeometry
21. GSModelInteriorTexture
22. GSModelInteriorDescriptor

- 23. GSModelInteriorMaterial
- 24. GSModelCMT
- 25. T2DModelGeometry
- 26. Navigation

All these datasets share the same 5-level directory structure defined below.

Table 3-28: Tiled Dataset Directory Structure

Director y Level	Directory Name	Description
Level 1	Lat	Geocell Latitude – This directory level divides the CDB along lines of latitude aligned to Geocells. By convention the name of the directory is based on the latitude of the south edge of the Geocell.
Level 2	Lon	Geocell Longitude – This directory level divides the CDB along lines of longitude aligned to Geocells. By convention the name of the directory is based on the longitude of the west edge of the Geocell.
Level 3	nnn_DatasetName	Tiled Dataset Name – The name of the directory is composed of the 3-digit dataset code (denoted nnn) followed by an underscore and the dataset name. Dataset codes are listed in Annex Q OGC CDB Core: Model and Physical Structure: Informative Annexes.
Level 4	LOD	This directory level divides each of the tiled datasets of the Geocell into its Level of Details
Level 5	UREF	This directory level divides a particular level of details into rows of tiles. UREF is a reference to the Up Index of a tile.

The above directory structure results in the following path to all files of the tiled datasets.

\CDB\Tiles\Lat\Lon\nnn_DatasetName\LOD\UREF\

Directory levels are further described below.

3.6.2.1 Directory Level 1 (Latitude Directory)

This section provides the algorithm to determine the name of the directory at level 1 of the Tiles hierarchy.

Requirement 65

<http://www.opengis.net/spec/cdb/1.0/core/latitude-directory-name>

The Latitude Directory naming *SHALL* implement the following policy. The directory name starts with either an “N” (North) for latitudes greater than or equal to 0 ($lat \geq 0$) or a “S” (South) for latitude less than 0 ($lat < 0$); this “N,S” prefix is followed by two digits:

if $lat < 0$ the directory name is “S($NbSliceID/2 - SliceID$)”

if $lat \geq 0$ the directory name is “N($SliceID - NbSliceID/2$)”

$SliceID$ and $NbSliceID$ are computed as per the following equations:

$$SliceID = \text{int}\left(\frac{lat + 90}{DlatCell}\right)$$

$$NbSliceID = 2 \times \text{int}\left(\frac{90}{DlatCell}\right)$$

where...

lat : is the latitude of the CDB tile

$DlatCell$: is the size in degree of a CDB Geocell in latitude

The CDB standard specifies $DlatCell$ to be 1 degree anywhere on earth, which gives 180 earth slices ($NbSliceID = 180$) and a $SliceID$ ranging from 0 to 179. Note that the latitude range of the CDB standard Earth Model Tiled Datasets is $-90 \leq lat < 90$; Refer to Section 2.1.3, Handling of the North and South Pole for the handling of the latitude of +90.

Note that the directory name corresponds to the latitude of the southwest corner of the CDB Geocell. Moreover, future releases of the CDB Specification shall retain the same value of $DlatCell$. Note that a modification of the value of $DlatCell$ would entail substantial changes to the resulting CDB directory and file naming thus requiring a re-compilation of existing CDBs.

3.6.2.1.1 Examples

Data elements located at latitude -5.2° will be found under the directory named:

`\CDB\Tiles\S06`

Data elements located at latitude $+62.3^\circ$ will be found under the directory named:

`\CDB\Tiles\N62`

3.6.2.2 Directory Level 2 (Longitude Directory)

This section provides the algorithm to determine the name of the directory at level 2 of the Tiles hierarchy.

Requirement 66

<http://www.opengis.net/spec/cdb/1.0/core/longitude-directory-name>

The Longitude Directory naming *SHALL* implement the following policy. The directory name prefix is “E” (East) for longitudes greater than or equal to 0 ($lon \geq 0$) and “W” (West) for longitudes less than 0 ($lon < 0$); three digits follow the prefix:

if $lon < 0$ the name is “W($NbSliceIDIndexEq/2 - SliceIDIndex$)”

if $lon \geq 0$ the name is “E($SliceIDIndex - NbSliceIDIndexEq/2$)”

$SliceIDIndex$ and $NbSliceIDIndexEq$ are computed as per the following equations:

$$SliceIDIndex = \text{int} \left(\text{int} \left(\frac{lon + 180}{DlonCell} \right) \times DlonZone(lat) \right)$$

$$NbSliceIDIndex = 2 \times \text{int} \left(\frac{180}{DlonCell} \right)$$

$$NbSliceIDIndexEq = 2 \times \text{int} \left(\frac{180}{DlonCellBasic} \right)$$

where...

lon is the longitude of the CDB tile

Note that $SliceIDIndex$ and $NbSliceIDIndex$ are a function of both latitude and longitude; however, $NbSliceIDIndexEq$ is the number of $SliceIDIndex$ at the equator. First, the longitude size of the CDB Geocell ($DlonCell$) is determined:

$$DLonCell = DLonCellBasic \times DLonZone(lat)$$

where...

DLonCellBasic is the width of a CDB Geocell in degrees at the equator

DLonZone(lat) is the number of *DLonCellBasic* in a given zone as per Table 3-29: NbSliceIDIndex for every CDB Zones. *DLonZone(lat)* is a function of the latitude.

The CDB standard sets *DLonCellBasic* to 1 degree, which gives 360 CDB Geocells (*NbSliceIDIndexEq*=360) at the equator. Table 3-29: NbSliceIDIndex for every CDB Zones, provides the values for *NbSliceIDIndex* at given latitudes. *SliceIDIndex* ranges from 0 to *NbSliceIDIndexEq*-1 at all latitudes. Note that the longitude range of the CDB Earth Model Tiled Datasets is $-180 \leq lon < 180$ which implies that an application needs to convert a longitude of 180 to -180 before computing *SliceIDIndex*.

Since *DLonCellBasic* is set to 1 degree, the index *SliceIDIndex* will increment by *DLonZone(lat)*; therefore, the directory name corresponds to the longitude of the southwest corner of the CDB Geocell. Moreover, future release of the CDB standard should retain the same value of *DLonCellBasic*. Doing otherwise will cause substantial modifications to the repository file naming convention and tile content thus requiring a conversion of the CDB instance.

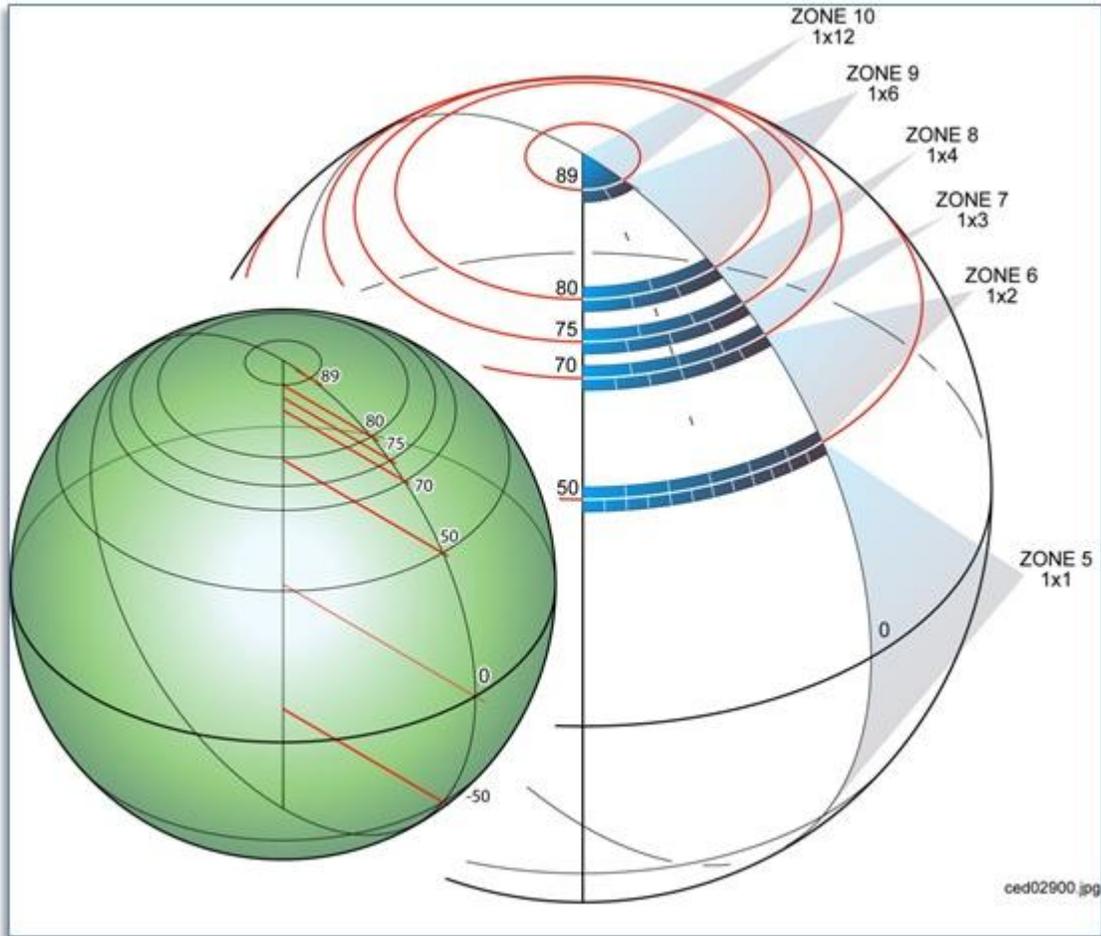


Figure 3- 8: Allocation of CDB Geocells with Increasing Latitude

Table 3-29: NbSliceIDIndex for every CDB Zones

Latitude	DLonZone(lat)	NbSliceIDIndex
$+89 \leq \text{lat} < +90$	12	30
$+80 \leq \text{lat} < +89$	6	60
$+75 \leq \text{lat} < +80$	4	90
$+70 \leq \text{lat} < +75$	3	120
$+50 \leq \text{lat} < +70$	2	180

$-50 \leq \text{lat} < +50$	1	360
$-70 \leq \text{lat} < -50$	2	180
$-75 \leq \text{lat} < -70$	3	120
$-80 \leq \text{lat} < -75$	4	90
$-89 \leq \text{lat} < -80$	6	60
$-90 \leq \text{lat} < -89$	12	30

3.6.2.2.1 Examples

Data elements located at latitude -5.2° and longitude $+45.2^\circ$ will be found under the directory named:

```
\CDB\Tiles\S06\E045
```

Data elements located at latitude $+62.3^\circ$ and longitude -160.4° will be found under the directory named:

```
\CDB\Tiles\N62\W162
```

The reason for “W162” instead of “W161” is that at latitudes between 50° and 70° , the CDB Geocells have a width of 2 degrees as indicated in Table 3-29: NbSliceIDIndex for every CDB Zones. “W162” corresponds to the southwest corner of the corresponding CDB Geocell.

3.6.2.3 Directory Level 3 (Dataset Directory)

The name of the directory at level 3 is composed of the dataset code and dataset name. The complete list is provided in Annex Q of the OGC CDB Core: Model and Physical Structure: Informative Annexes. Examples are provided below.

3.6.2.3.1 Examples

The elevation and the imagery of the geocell located at a latitude of -6° and a longitude of $+45^\circ$ will be found under the directories named:

```
\CDB\Tiles\S06\E045\001_Elevation
```

```
\CDB\Tiles\S06\E045\004_Imagery
```

The list of geospecific features of the geocell located at latitude $+62^\circ$ and longitude -160° will be found under the directory named:

```
\CDB\Tiles\N62\W160\100_GSFeature
```

The network of roads covering the geocell located at latitude $+62^\circ$ and longitude -160° will be found under the directory named:

```
\CDB\Tiles\N62\W160\201_RoadNetwork
```

The geometry and textures of a geospecific 3D model located at latitude -5.2° and longitude $+45.2^\circ$ will be found under the directories named:

```
\CDB\Tiles\S06\E045\300_GSModelGeometry
```

`\CDB\Tiles\S06\E045\301_GSModelTexture`

The geometry of tiled 2D models covering the geocell located at latitude -5° and longitude $+45^\circ$ will be found under the directories named:

`\CDB\Tiles\S05\E045\310_T2DModelGeometry`

To complete these examples, the files associated with the Navigation dataset and covering the geocell located at latitude $+36^\circ$ and longitude -88° will be found under the directory named:

`\CDB\Tiles\N36\W088\401_Navigation`

3.6.2.4 Directory Level 4 (LOD Directory)

This directory level contains all of the Level of Details directories supported by the corresponding datasets.

All coarse LOD tiles, ranging from LOD -10 to LOD -1 , are stored in a single directory uniquely named `\LC`. The remaining finer LODs (i.e., LOD 0 to LOD 23) have their own corresponding directories, named `\Lxx` where `xx` is the 2-digit LOD number.

3.6.2.4.1 Examples

LOD 2 of the terrain elevation of the geocell located at latitude -6° and longitude $+45^\circ$ will be found under the directory named:

`\CDB\Tiles\S06\E045\001_Elevation\L02`

LOD -6 of the same dataset for the same geocell will be found in:

`\CDB\Tiles\S06\E045\001_Elevation\LC`

3.6.2.5 Directory Level 5 (UREF Directory)

The UREF directory level subdivides a geocell into a number of rows to limit the number of entries in a directory.

The number of files at a given LOD is proportional to $2^{2 \times \text{LOD}}$. For instance, LOD 10 represents about one million files. The introduction of the UREF directory level reduces the number of files per directory to the order of 2^{LOD} .

The name of the directory is composed of the character U (Up direction) followed by the Up index (or the row number) of the tile, as described in this section.

The number of rows in a CDB Geocell at a given LOD is given by the following equation:

$$U_{NRowLod} = \max \left(\text{int} \left(\frac{2^{Lod+10}}{2^{10}} \right), 1 \right)$$

...which simplifies to:

$$U_{NRowLod} = \max(2^{Lod}, 1)$$

The index of a row ranges from 0 for the bottom row to $U_{NRowLod}-1$ for the upper row. For any given latitude lat , its Up Index U_{Ref} is determined by first computing $DLat$:

$$DLat = (lat + 90) - \text{int}\left(\frac{lat + 90}{DLatCell}\right) \times DLatCell$$

...which simplifies to the following for computer language that support modulo:

$$DLat = \text{mod}(lat + 90, DLatCell)$$

Then the index of the UREF can be evaluated as follows:

$$U_{Ref} = \text{int}\left(\frac{DLat \times 2^{Lod}}{DLatCell}\right)$$

Knowing that the value of $DLatCell$ is 1° for the whole CDB, the resulting formulas become:

$$\underline{DLat} = \text{mod}(lat + 90, 1)$$

$$U_{Ref} = \text{int}(\underline{DLat} \times 2^{LOD})$$

3.6.2.5.1 Examples

At latitude -5.2° and at LOD 2, the UREF index computed from the above formulas will be:

$$DLat = \text{mod}(-5.2 + 90, 1) = \text{mod}(84.8, 1) = 0.8$$

$$U_{Ref} = \text{int}(0.8 \times 2^2) = \text{int}(0.8 \times 4) = \text{int}(3.2) = 3$$

Assuming longitude $+45.2^\circ$, the elevation data corresponding to this coordinate will be found under the directory named:

\CDB\Tiles\S06\E045\001_Elevation\L02\U3

A geospecific feature whose significant size qualifies it for LOD 7 and positioned at latitude +62.3° will produce the following UREF index:

$$DLat = \text{mod}(62.3 + 90, 1) = \text{mod}(152.3, 1) = 0.3$$

$$U_{Ref} = \text{int}(0.3 \times 2^7) = \text{int}(0.3 \times 128) = \text{int}(38.4) = 38$$

Assuming longitude -160.4°, the data will be found under the directory named:

\CDB\Tiles\N62\W162\100_GSFeature\L07\U38

3.6.3 Tiled Dataset File Naming Conventions

There are two sets of naming conventions for tiled datasets. The first one corresponds to the name of files located in the leaf directories of the **\CDB\Tiles** hierarchy. The second set of names applies to files found inside ZIP archives.

3.6.3.1 File Naming Convention for Files in Leaf Directories (UREF Directory)

Requirement 67

<http://www.opengis.net/spec/cdb/1.0/core/uref-directory-name>

All files stored in the UREF subdirectory of section 3.6.2.5 *SHALL* have the following naming convention: LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn.xxx

The following table defines each field of the file name and chapter 5, CDB Datasets, provides the dataset codes and the component selectors to complete the name.

Table 3-30: Tiled Dataset File Naming Convention 1

Field	Description
Lat	Geocell Latitude – Identical to the name of the directory defined in section 3.6.2.1, Directory Level 1 (Latitude Directory).
Lon	Geocell Longitude – Identical to the name of the directory defined in section 3.6.2.2, Directory Level 2 (Longitude Directory).
Dnnn	Character D followed by the 3-digit code assigned to the dataset.

Snnn	Character S followed by the 3-digit value of Component Selector 1.
Tnnn	Character T followed by the 3-digit value of Component Selector 2.
LOD	Level of Detail – As defined in section 3.3.8.5, Level of Detail.
Un	UREF – Identical to the name of the directory as defined in section 3.6.2.5, Directory Level 5 (UREF Directory).
Rn	RREF – A reference to the Right Index of a tile. Character R (Right direction) followed by the column number as described in this section.
xxx	File extension as per file type.

The RREF token divides a particular level of details into columns of tiles. The number of columns in a CDB Geocell at a given LOD is given by the following equation:

$$R_{NCollod} = \max\left(\text{int}\left(\frac{2^{Lod+10}}{2^{10}}\right), 1\right)$$

...which simplifies to:

$$R_{NCollod} = \max(2^{Lod}, 1)$$

The index of a column ranges from 0 for the leftmost column to $R_{NCollod}-1$ for the rightmost column. For any given *lat/lon* coordinate, its Right Index R_{Ref} is determined by first computing $DLon$:

$$DLon = (lon + 180) - \text{int}\left(\frac{lon + 180}{DLonCell}\right) \times DLonCell$$

...which simplifies to the following for computer language that support modulo:

$$DLon = \text{mod}(lon + 180, DLonCell)$$

Then the Right Index R_{Ref} can be evaluated as follows:

$$R_{Ref} = \text{int} \left(\frac{D_{Lon} \times 2^{Lod}}{D_{LonCell}} \right)$$

By substituting *DLonCell* that is defined in section 3.6.2.2, Directory Level 2 (Longitude Directory), we obtain the following set of equations:

$$D_{Lon} = \text{mod}(\text{lon} + 180, D_{LonZone}(\text{lat}))$$

$$R_{Ref} = \text{int} \left(\frac{D_{Lon} \times 2^{Lod}}{D_{LonZone}(\text{lat})} \right)$$

The value of *DLonZone* is provided by Table 3-29: NbSliceIDIndex for every CDB Zones.

3.6.3.1.1 Examples

Continuing from the examples in section 3.6.2.5.1, at latitude -5.2° and longitude $+45.2^\circ$ and at LOD 2, the RREF index computed from the above formulas will be:

$$D_{Lon} = \text{mod}(45.2 + 180, D_{LonZone}(-5.2)) = \text{mod}(225.2, 1) = 0.2$$

$$R_{Ref} = \text{int} \left(\frac{0.2 \times 2^2}{D_{LonZone}(-5.2)} \right) = \text{int} \left(\frac{0.2 \times 4}{1} \right) = \text{int}(0.8) = 0$$

The primary elevation data corresponding to this coordinate will be found in the file named:

S06E045_D001_S001_T001_L02_U3_R0.tif

A man-made point feature whose significant size qualifies it for LOD 7, positioned at latitude $+62.3^\circ$ and longitude -160.4° will produce the following RREF index:

$$D_{Lon} = \text{mod}(-160.4 + 180, D_{LonZone}(62.3)) = \text{mod}(19.6, 2) = 1.6$$

$$R_{Ref} = \text{int} \left(\frac{1.6 \times 2^7}{D_{LonZone}(62.3)} \right) = \text{int} \left(\frac{1.6 \times 128}{2} \right) = \text{int}(102.4) = 102$$

If ShapeFiles are being used, this approach results in the following file names:

N62W162_D100_S001_T001_L07_U38_R102.shp
 N62W162_D100_S001_T001_L07_U38_R102.shx
 N62W162_D100_S001_T001_L07_U38_R102.dbf
 N62W162_D100_S001_T001_L07_U38_R102.dbt

3.6.3.2 File Naming Convention for Files in ZIP Archives

The following GSModel datasets reside inside ZIP archives.

1. GSModelGeometry
2. GSModelTexture
3. GSModelMaterial
4. GSModelDescriptor
5. GSModelInteriorGeometry
6. GSModelInteriorTexture
7. GSModelInteriorMaterial
8. GSModelInteriorDescriptor

Requirements Class - Archive Names (68-72)	
/req/core/tiled-data	
Target type	Operations
Dependency	Various XML schema
Requirement 68	/req/core/archive-names
Requirement 69	/req/core/gsmodelgeometry-archive-name
Requirement 70	/req/core/gsmodeltexture-archive-name
Requirement 71	/req/core/msmodelmaterial-archive-name
Requirement 72	/req/core/gsmodeldescriptor-archive-name

Requirement 68
http://www.opengis.net/spec/cdb/1.0/core/archive-names

These files are stored in archives whose names *SHALL* follow the naming convention defined in section 3.6.3.1 above; the files inside those archives *SHALL* follow the naming conventions defined here: LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_ "*extra_tokens*".xxx

The extra tokens are described in the next sections.

3.6.3.2.1 *GSMoel Geometry File Naming Conventions*

Requirement 69

<http://www.opengis.net/spec/cdb/1.0/core/gsmoelgeometry-archive-name>

The files from the GSMoelGeometry and GSMoelInteriorGeometry datasets *SHALL* have the following naming convention:

LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_FeatureCode_FSC_MODL.flr

The FeatureCode, FSC, and MODL tokens are as defined in section 3.3.8.1, Feature Classification, and section 3.3.8.2, Model Name.

3.6.3.2.2 *GSMoel Texture File Naming Conventions*

Requirement 70

<http://www.opengis.net/spec/cdb/1.0/core/gsmoeltexture-archive-name>

The files from the GSMoelTexture and GSMoelInteriorTexture datasets *SHALL* have the following naming convention: LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_TNAM.rgb

The TNAM token is as defined in section 3.3.8.4, Texture Name.

3.6.3.2.3 *GSMoel Material File Naming Conventions*

Requirement 71

<http://www.opengis.net/spec/cdb/1.0/core/msmoelmaterial-archive-name>

The files from the GSMoelMaterial and GSMoelInteriorMaterial datasets *SHALL* have

the following naming convention: LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_TNAM.tif

The TNAM token is as defined in section 3.3.8.4, Texture Name.

3.6.3.2.4 *GSMModel Descriptor File Naming Conventions*

Requirement 72

<http://www.opengis.net/spec/cdb/1.0/core/gsmodeldescriptor-archive-name>

The files from the GSMModelGeometry and GSMModelInteriorGeometry datasets *SHALL* have the following naming convention:

LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_FeatureCode_FSC_MODL.flt

The FeatureCode, FSC, and MODL tokens are as defined in section 3.3.8.1, Feature Classification, and section 3.3.8.2, Model Name.

3.6.3.2.5 *Examples*

All archives at LOD 7 that are located at latitude +62.3° and longitude -160.4° will be named:

N62W162_Dnnn_S001_T001_L07_U38_R102.zip

For each model dataset that uses an archive, the name of the archive will be:

N62W162_D300_S001_T001_L07_U38_R102.zip (Geometry)
N62W162_D301_S001_T001_L07_U38_R102.zip (Texture)
N62W162_D302_S001_T001_L07_U38_R102.zip (Signature)
N62W162_D303_S001_T001_L07_U38_R102.zip (Descriptor)
N62W162_D304_S001_T001_L07_U38_R102.zip (Material)
N62W162_D305_S001_T001_L07_U38_R102.zip (Interior Geometry)
N62W162_D306_S001_T001_L07_U38_R102.zip (Interior Texture)
N62W162_D307_S001_T001_L07_U38_R102.zip (Interior Descriptor)
N62W162_D308_S001_T001_L07_U38_R102.zip (Interior Material)

Examples of files found inside the above archives could be:

N62W162_D300_S001_T001_L07_U38_R102_AL015_116_AcmeFactory.flt
N62W162_D301_Snnn_Tnnn_L07_U38_R102_AcmeFactory.rgb

```

N62W162_D302_Snnn_Tnnn_L07_U38_R102_AL015_116_AcmeFactory.shp
N62W162_D302_Snnn_Tnnn_L07_U38_R102_AL015_116_AcmeFactory.shx
N62W162_D302_Snnn_Tnnn_L07_U38_R102_AL015_116_AcmeFactory.dbf
N62W162_D303_S001_T001_L07_U38_R102_AL015_116_AcmeFactory.xml
N62W162_D304_Snnn_Tnnn_L07_U38_R102_AcmeFactory.tif
N62W162_D305_S001_T001_L07_U38_R102_AL015_116_AcmeFactory.flt
N62W162_D306_S001_T001_L07_U38_R102_AcmeFactoryWall.rgb
N62W162_D306_S001_T001_L07_U38_R102_AcmeFactoryFloor.rgb
N62W162_D306_S001_T001_L07_U38_R102_AcmeFactoryCeiling.rgb
N62W162_D307_S001_T001_L07_U38_R102_AL015_116_AcmeFactory.xml
N62W162_D308_Snnn_Tnnn_L07_U38_R102_AcmeFactory.tif

```

To complete the example, the name of the GSModel Composite Material Table (GSModelCMT) that is associated with the above geocell is:

```
N62W162_D309_S001_T001_L00_U0_R0.xml
```

Note that this file is located at LOD 0, hence the value 0 for UREF and RREF. This will be explained later in chapter 5.

Finally, the geometry of the tiled 2D model corresponding to the above tile will be named:

```
N62W162_D310_S001_T001_L07_U38_R102.flt
```

3.7 Navigation Library Dataset

The `\CDB\Navigation\` folder is the root directory of the Navigation library which is composed of a single dataset.

1. NavData

The purpose of the Navigation library is to include all of the information which is either not geographically located, or has a global geographical coverage and can be used as a lookup to the Navigation Tile-LODs.

3.7.1 Navdata Structure

The NavData dataset is assigned dataset code 400 and has a single level directory structure.

Table 3-31: GTModelGeometry Entry File Directory Structure

Director y Level	Directory Name	Description
Level 1	400_NavDat	The name of the directory is composed of the dataset

	a	code followed by an underscore and the dataset name.
--	---	--

3.7.2 Navigation Data Naming Convention

Requirement 73
http://www.opengis.net/spec/cdb/1.0/core/navdata-naming
All files of the NavData dataset <i>SHALL</i> have the following naming convention: D400_Snnn_Tnnn.dbf

The following table defines each field of the file name and section 5.2 provides the values of the Component Selectors to complete the name.

Table 3-32: NavData Naming Convention

Field	Description
D400	Character D followed by the 3-digit code assigned to the dataset.
Snnn	Character S followed by the 3-digit Component Selector 1
Tnnn	Character T followed by the 3-digit Component Selector 2
dbf	The file type associated with the dataset (for dBASE III+ file or database table)

3.7.2.1 Examples

The Schema file (T002) of the Airport component (S001) of the NavData dataset is stored in:

```
\CDB\Navigation\400_NavData\D400_S001_T002.dbf
```

4. CDB File Formats

The CDB standard internal formats are based on the formats used by industry-standard toolsets. This approach eliminates the time-consuming off-line data assembly and data publishing process usually imposed by each of the clients. Refer to Section 1.6.4.2, Data Store Generation Flow for a more comprehensive discussion of this topic.

Furthermore, the translation step into a CDB specified storage format is typically trivial since the CDB standard is based on industry-standard native tool formats. Please note that a CDB

structured data store supports other file types. For example, an OGC GeoPackage file could be stored in the CDB structure. However, a compliance test may throw an exception stating that the GeoPackage file type is not recognized.

The CDB standard permits any CDB to run “as-is”, without any offline assembly (aka compilation), translation, conversion, on any CDB-compliant simulator client-device platform. This allows the simulator user community AND the database creation community to freely exchange CDBs across simulators and database generation facilities either through the exchange of physical media (or entire storage subsystems) or via network. As a result, a CDB structured data store can be run and exchanged without change on any CDB-compliant simulator client-devices or any database generation workstations, regardless of the computer platforms, simulator system software.

The storage structure of the CDB standard allows for efficient searching, retrieval and storage of any information contained within the data store. The storage structure portion of the CDB data model is defined in Chapter 3.

The formats currently used in a CDB compliant data stores are:

1. **TIFF (*.tif):** used for the representation of all datasets whose inherent structure reflects that of a two-dimensional regular grid in a Cartesian coordinate system. The primary use of TIFF within a CDB conformant data store is for the representation of terrain elevation and raster imagery. To qualify as a CDB-compliant TIFF reader, the reader must satisfy the requirements described in Clause 8 of Volume 10: OGC CDB Implementation Guidance. Please note that the LZW compression algorithm within the TIFF format is supported and encouraged by the CDB standard when the data type of the content of the file is of integral type. As a consequence, it is strongly recommended to compress TIFF files containing integer values but to avoid compression if the file contains floating-point values.
2. **GeoTIFF (*.tif):** used for the representation of all datasets whose inherent structure reflects that of a two-dimensional regular grid of a Geographic coordinate system. The primary use of GeoTIFF within a CDB data store is for the representation of terrain elevation (note: the use GeoTIFF is preferred over TIFF in the case of terrain elevation). CDB-compliant GeoTIFF readers do not concern themselves with any of the GeoTIFF specific tags because the CDB standard provides all of the conventions to geo-reference each geographic dataset. However, it is strongly recommended that data store generation tools be fully compliant to GeoTIFF; this provision eliminates the need for the tools to be aware of the CDB conventions governing the content of each geo-referenced dataset.
3. **SFI Format (*.rgb):** used for the representation of 3D model textures. The file format allows for the representation of an image with 1, 2, 3, or 4 channels. A single channel image represents a grey-shaded texture; a two-channel image represents a grey-shaded texture with an alpha component providing the transparency; a three-channel image represents a color (RGB) texture; finally, a four-channel image is a color (RGB) texture with an alpha channel providing the transparency. CDB-compliant RGB readers must be

fully compliant with the SGI Image File Format Specification. Its use is limited to 3D models.

4. **JPEG 2000 (*.jp2):** used for the representation of an image encoded in accordance to the JPEG 2000 standard. CDB-compliant JPEG 2000 readers must be fully compliant with the JPEG 2000 standard while reading such still image file types. JPEG 2000 encoded images can be used for the representation of geo-referenced terrain imagery with some degree of compression levels and is only applicable in the case of terrain raster imagery. Volume 2 CDB Core: Model and Physical Structure: Annexes C and H describe the use of the JPEG 2000 file format in a CDB data store.
5. **3D Models:** Currently OpenFlight is used for the representation of 3-dimensional geometric representation of all models (statically positioned cultural models, moving models). Please refer to the requirements described in Volume 6 OGC CDB Rules for Encoding Data using OpenFlight <need URL>.
6. **Vector features and geometry:** Currently ShapeFiles are used for the representation and attribution of the vector feature datasets in a CDB data store. Please see the requirements described in Volume 4: OGC CDB Best Practice use of Shapefiles for Vector Data Storage.
7. **Extensible Markup Language (*.xml):** used to store metadata that describes CDB versioning, describes CDB Composite and Base material structure, defines CDB light type naming conventions and hierarchy, and defines CDB model component hierarchy.
8. **Cross-platform and interoperable file storage and transfer format (*.zip):** used to archive and store geospecific 3D model datasets.

For all other formats (used by this standard), CDB readers should be fully compliant²⁹.

File Format	Versions Currently Supported by CDB	CDB Client-device Behavior for Prior Versions
*.tif	6.0	Ignores data
*.rgb	1.0	Ignores data
*.jp2	1.0	Ignores data
*.flt	16.0	Ignores data
*.shp, *.shx	Esri White Paper, July 98	Ignores data
*.dbf, *.dbt	dBASE III+ and above	Ignores data
*.xml	1.0	Ignores data
*.zip	6.3.1	Ignores data

²⁹ This table will be expanded as Best Practices for additional formats and encodings are developed.

5. CDB Datasets

This chapter provides the description of the content of datasets in a CDB data store, except OpenFlight and RCS models that are covered in separate documents (Volumes 5 and 6 of this standard). Chapter 5 also provides the Component Selectors necessary to complete the file names associated with all CDB datasets

5.1 Metadata Datasets

Metadata datasets contain information that is global to a CDB data store and that define its structure, naming hierarchies, default values, allowable values, and status. All metadata files are formatted using eXtended Markup Language (XML) files and their XML schemas can be found in the `\CDB\Metadata\Schema\` folder delivered with the CDB standard.

Requirements Class - Metadata Datasets (74-80)	
/req/core/metadata-datasets	
Target type	Operations
Dependency	Various XML schema
Requirement 74	/req/core/version
Requirement 75	/req/core/attribute-definition
Requirement 76	/req/core/attribute-schema-level
Requirement 77	/req/core/attribute-value
Requirement 78	/req/core/attribute-units
Requirement 79	/req/core/attribute-scaler
Requirement 80	/req/core/ao1

The table below lists all metadata files that are allowed and defined by the CDB standard. Note that a dataset code and component selectors are assigned to each metadata file even though these codes do not participate in the construction of their file names. Dataset codes are assigned to metadata datasets for consistency with all other CDB Datasets.

Table 5-1: Component Selectors for Metadata Datasets

CS1	CS2	File Name
Dataset 700, Metadata		

001	-	Lights.xml
002	-	Model_Components.xml
003	-	Materials.xml
004	-	Defaults.xml
005	-	Specification_Version.xml (Deprecated)
006	-	Version.xml
007	-	CDB_Attributes.xml
008	-	Geomatics_Attributes.xml
009	-	Vendor_Attributes.xml
010	-	Configuration.xml
Dataset 701, Client-Specific Metadata		
-	-	Lights_xxx.xml

For client-specific metadata, the standard only reserves one dataset code but no component selector. The mechanism is kept for backward compatibility with previous versions of the CDB standard. However, its use is strongly discouraged because it defeats the very intent of the CDB, which is to promote correlation between client devices by having a single source of data.

5.1.1 Light Data Hierarchy Metadata

The light name hierarchy for a CDB compliant data store is described in detail within the table found in Volume 2 OGC CDB Core Model and Physical Structure Annexes - Annex J - of this standard. For run-time access of this data, clients need to retrieve such information. To this end, the Lights Hierarchy Definition metadata is stored in an XML file in the metadata CDB directory as described in Section 3.1.1, Metadata Directories. The name of the file is “**Lights.xml**”.

The XML file provides a description of the entire naming hierarchy, including the hierarchical relationship of the levels with respect to each other and the position of each light type within this hierarchy. In addition to the name of each light type, the “**Lights.xml**” file contains a unique code with each light type.

In the case of light features (Airport Features - Lights and Environmental Lights tiled datasets), the light type code provides a storage-efficient means to attribute each light, since only the code is used to attribute light features. Data processing tools are required to map the light type name string provided by the modeler into a light type code.

NOTE: In the case of light features that are part of OpenFlight models, the light type name string provided by the modeler is used “as-is” within the model to attribute each of the light features.

Client-devices are required to internally build and initialize a table of light properties and characteristics for their respective use. This table could be indexed at runtime using the light type code. The table can be built at CDB data store load time and should match the device’s inherent capabilities and level-of-fidelity; this flexibility can be achieved because the “**Lights.xml**” file communicates the lights naming hierarchy to the client-devices.

The client-devices are required by the CDB standard to ensure that properties and characteristics of lower-tier names in the light point hierarchy inherit the properties and characteristics of the higher-tier names in the light name hierarchy. This feature allows modelers to add new light names to the light name hierarchy and be assured that the new light names will immediately inherit all of the properties and characteristics of the parent names even if the simulator vendor does not update any of the client-devices.

The light type code can range from **0 to 9,999**. The light type codes are used by the Airport Features - Lights and Environmental Lights tiled datasets of the CDB. It is up to the CDB creation tools to ensure that the light type code does in fact correspond to the light type name assigned by the modeler.

Below is a small sample of the CDB light name hierarchy in XML format.

```

<Lights>
  <Light type="Light" code="0">
    <Description>
      All Purpose Generic light
    </Description>
  <Light type="Platform" code="1">
    <Description>
      Platform light
    </Description>
  <Light type="Air" code="2">
    <Description>
      Aircraft light
    </Description>
  <Light type="Aircraft_Helos" code="3">
    <Description>
      Light for Aircraft and Helicopters
    </Description>
  <Light type="Anti-collision" code="4">
    <Description>
      Anti collision light - normally red flashing
    </Description>
  <Light type="Bottom_Light" code="5">
    <Description>
      Anti-collision found on bottom of the fuselage
    </Description>
  <Light type="NVG_Bottom_Light" code="6">
    <Description>
      Anti-collision found on bottom of fuselage in NVG mode
    </Description>
  </Light>
</Light>
  ... other light definitions of type Platform-Air-Aircraft_Helos
</Light>
  ... other light definitions of type Platform-Air
</Light>
  ... other light definitions of type Platform
</Light>
</Light>
</Lights>

```

Note that light code numbering need not be consecutive. Light codes have a one-to-one association with light types; consequently, the light codes are unique among all light types.

5.1.1.1 Client Specific Lights Definition Metadata

Client-devices use the light type code as an index to lookup the client-specific properties and characteristics of each light type. This approach is client-device independent because the (device-specific) client's rendering parameters are local to its implementation. As a result, modelers need not bother setting or even understanding the many parameters specific to each light type and to each client-device type.

The CDB standard also offers a complementary approach to modifying the appearance of lights. This approach provides basic control over light intensity, color, lobe width and aspect, frequency and duty cycle to client devices. The approach also permits a modeler to add new light types to the CDB light hierarchy.

Example:

As an example, we will create a client-specific lights definition metadata file for a hypothetical client-device. The information would be held in the Lights_xxx.xml metadata file corresponding to the client-device for which lights are to be tuned. There can be one file per client-device and the file for each client-device is optional. The file is not required if the modeler does not wish to adjust the basic characteristics of one or more light types for the associated client-device, or he/she doesn't require new light types to be added to the CDB light hierarchy. The metadata file would be loaded by the client-device whose name matches the "xxx" character string of the Lights_xxx.xml file. As for the Lights, the file would be located at the top of the CDB storage hierarchy in directory \CDB\Metadata\ as described in Section 3.1.1, Metadata Directories.

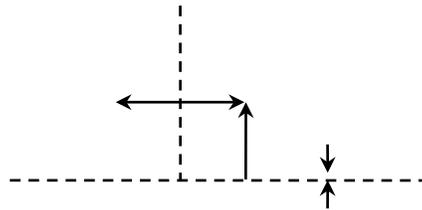
Nominally, the Lights_xxx.xml consists of light type entries corresponding to the light types the modeler wishes to add/modify. Each entry in the Lights_xxx.xml file consists of one or optional fields.

Consider the case of a simulator equipped with a client-device rendering simulated imagery for model "A" NVG goggles and a second client-device rendering simulated imagery for model "B" NVG goggles. After viewing the CDB on the simulator, the modeler wishes to diminish the intensity of the \Lights\Cultural\Line-based\Highway lights for model "A" NVG goggles to 90% of the intensity calculated by the simulator. To do this, the modeler creates a Lights_NVG_A.xml, creates a light type entry for \Lights\Cultural\Line-based\Highway and provides an intensity field with value of 0.9. Note that all other characteristics of the light type in this client-device are unaffected since the modeler did not provide additional fields. Furthermore, the characteristics of the light type in all other client-devices remain unaffected since the modeler did not provide other Lights_xxx.xml files.

The XML schema for the fields of the Lights_xxx.xml is delivered with the standard in \CDB\Schema\Lights_Tuning.xsd. The fields are:

- **Intensity:** When a light type is non-native to the CDB standard, which means that it is without a corresponding entry in Annex J Intensity represents the light point intensity for the client-device (range normalized from 0.0 to 1.0). When the light entry is native to the CDB standard, Intensity is used as a floating-point intensity modifier that multiplies the intensity calculated by the client-device. In both cases, Intensity defaults to a value of 1.0.
- **Color:** When a light type is non-native to the CDB standard, Color is a floating-point RGB triplet that represents the color of the light type for the client-device (range normalized from 0.0 to 1.0). When the light entry is native to the CDB specification, Color is a floating-point RGB triplet that multiplies the RGB value calculated by the client-device. Color applies only to visual system client-device types. If absent in a light type entry, Color defaults to a value of white (1.0, 1.0, 1.0).
- **Directionality:** A string that categorizes the light type as “Omnidirectional”, “Directional” or “Bidirectional”. If absent in a light type entry, Directionality defaults to the value “Omnidirectional”.
- **Lobe_Width:** Represents the identifying section for the light’s lobe width characteristics, which can have a horizontal and vertical attribute.

- **Horizontal:** When a light type is non-native to the CDB specification, the Horizontal field represents the light point’s half-intensity horizontal lobe width for the client-device (range from 0.0 to 360.0). When the light entry is native to the CDB standard, Horizontal field is used as a floating-point modifier that multiplies the horizontal lobe width calculated by the client-device. Applies only to Directional and Bidirectional light types. If absent in a light type entry, Horizontal field defaults to a value of 1.0.



- **Vertical:** When a light type is non-native to the CDB standard, Vertical field represents the light point’s half-intensity vertical lobe width for the client-device (range from 0.0 to 360.0). When the light entry is native to the CDB standard, Vertical field is used as a floating-point modifier that multiplies the vertical lobe width calculated by the client-device. This applies only to Directional and Bidirectional light types. If absent in a light type entry, Vertical field defaults to a value of 1.0.

- **Residual_Intensity:** When a light type is non-native to the CDB standard, Residual_Intensity represents the residual intensity of the light. Residual intensity is the intensity of the light (range normalized from 0.0 to 1.0) outside of the lobe defined by Lobe_Width:Horizontal and Lobe_Width:Vertical fields. When the light entry is native to the CDB specification, Residual_Intensity is used as a

floating-point modifier that multiplies the residual intensity calculated by the client-device. This applies only to Directional and Bidirectional light types. If absent in a light type entry, Residual_Intensity defaults to a value of 1.0.

- **Frequency:** A floating-point value greater than or equal to 0.0 that sets the blink or rotating frequency of the light in Hertz (cycles per second). A value of 0.0 disables all blinking and rotating properties. If absent in a light type entry, Frequency defaults to a value of 0.0.
- **Duty_Cycle:** A floating-point value ranging from 0.0 to 1.0 that sets the duty cycle of the light. Duty cycle is defined as the percentage of time the light is turned on over a complete cycle. A value of 0.0 permanently turns the light off. A value of 1.0 turns it on. The value is ignored if Frequency = 0.0. If absent in a light type entry, Duty_Cycle defaults to a value of 0.5.

Here is a sample of a *Lights_xxx.xml* file where a modeler has exercised explicit control over the properties of an anti-collision light and a landing light.

```
<Lights_Tuning>
  <Light type="\Light\Platform\Air\Aircraft_Helos\Anti-collision">
    <Description>Tuned for MH-47 CMS</Description>
    <Intensity>0.75</Intensity>
    <Color>1.0 0.0 0.0</Color>
    <Directionality>Omnidirectional</Directionality>
    <Frequency>0.5</Frequency>
    <Duty_Cycle>0.2</Duty_Cycle>
  </Light>
  <Light type="\Light\Platform\Air\Aircraft_Helos\Landing">
    <Description>...</Description>
    <Intensity>1.0</Intensity>
    <Color>1.0 0.9 0.9</Color>
    <Directionality>Directional</Directionality>
    <Residual_Intensity>0.05</Residual_Intensity>
    <Lobe_Width>
      <Horizontal>30.0</Horizontal>
      <Vertical>25.0</Vertical>
    </Lobe_Width>
  </Light>
</Lights_Tuning>
```

5.1.2 Model Components Definition File

The CDB standard provides the means to unambiguously tag any portions of a 3D model (moving model or cultural feature with a modeled representation) with a descriptive name. Component model names are stored in the model components definition file, “\CDB\Metadata\Model_Components.xml” as described in Section 3.1.1, Metadata Directories. The XML file containing the CDB Model Components is part of the CDB standard distribution package. The XML schema is provided in \CDB\Metadata\Schema\Model_Components.xsd delivered with the standard.

The following shows a content sample of the model component definition file:

```
<Model_Components>
  <Component name="Artillery_Gun">
    <Description>
      1) Refers to any engine used for the discharge of large
         projectiles and served by a crew of men.
      2) Cannon-like weapons operated by more than one person.
    </Description>
  </Component>
  <Component name="Windshield">
    <Description>
      A transparent screen located in front of the occupants of a
      vehicle to protect them from the wind and weather.
    </Description>
  </Component>
  ... other components
</Model_Components>
```

5.1.3 Base Materials Table

CDB Base Materials are listed in \CDB\Metadata\Materials.xml and stored in an XML file named \CDB\Metadata\Materials.xml, as mentioned in section 3.1.1. The format of the file is defined by an XML schema that is delivered with the CDB standard in the file named \CDB\Metadata\Schema\Base_Material_Table.xsd.

Here is an excerpt of the CDB Base Material Table showing the definitions of the first and the last base materials of the standard.

```

<Base_Material_Table>
  <Base_Material>
    <Name>BM_ASH</Name>
    <Description>
      The solid remains of a fire
    </Description>
  </Base_Material>
  ...
  <Base_Material>
    <Name>BM_WOOD-DECIDUOUS</Name>
    <Description>
      Trunks, branches of live deciduous trees
    </Description>
  </Base_Material>
</Base_Material_Table>

```

5.1.4 Default Values Definition Table

Default values for all datasets can be stored in the default values metadata file “\CDB\Metadata\Defaults.xml” as described in Section 3.1.1, Metadata Directories. Default values defined throughout the CDB standard are listed in Annex S OGC CDB Core: Model and Physical Structure: Informative Annexes. The XML schema is provided in \CDB\Metadata\Schema\Defaults.xsd delivered with the standard. There are two types of default values: read and write default values (‘R’ or ‘W’.) Generally, read default values are values to be used when optional information is not available. Write default values are default values to be used by CDB creation tools to fill mandatory content when information is either missing or not available. The default value name is a unique name identifying a default value for a given dataset. Valid default value names are listed in Annex S. Each default value has a type. Valid default value data types are “float”, “integer” and “string”.

The following is an excerpt of a “Defaults.xml” file containing the default terrain elevation value.

```

<Default_Value_Table>
  <Default_Value>
    <Dataset>001_Elevation</Dataset>
    <Name>Default_Elevation-1</Name>
    <Description>Default Primary Terrain Elevation</Description>
    <Type>float</Type>
    <Value>0.0</Value>
    <R_W_Type>R</R_W_Type>
  </Default_Value>
  <!-- Insert other Default Values in accordance to the table above -->
</Default_Value_Table>

```

5.1.5 No content

5.1.6 Version Metadata

Requirement 74

<http://www.opengis.net/spec/cdb/1.0/core/version>

Each CDB Version *SHALL* have a version control file that is called Version.xml. Its contents are as follows:

```

<Version>
  <PreviousIncrementalRootDirectory name="Path to another CDB Version"/>
  <Comment>A comment to describe this CDB Version</Comment>
  <Specification version="3.2|3.1|3.0"/>
  <Extension name="name of the extension" version="version of this extension"/>
</Version>

```

The complete XML schema can be found in \CDB\Metadata\Schema\Version.xsd delivered with the standard.

The optional <PreviousIncrementalRootDirectory> element is used to refer to another CDB Version. This is the mechanism to use to chain together two CDB versions. The optional <Comment> element is a free-format text to describe the purpose and/or the nature of the data of this CDB Version. The mandatory <Specification> element indicates the CDB standard that is used to produce the content of this CDB Version. Note that version numbers of the standard are limited to the existing versions: 3.2, 3.1, and 3.0. Other values are not permitted. Finally, the optional <Extension> element indicates that this CDB Version is in fact a CDB Extension.

A version control file that does not have a CDB Extension indicates that the CDB Version holds content that strictly follows the CDB standard.

A CDB Extension corresponds to user defined information, which is not described or supported by the CDB standard, stored within the CDB Version. As an example, such additional information could be client or vendor-specific information used to increase system performance. Any user defined information shall not replace or be used in place of existing CDB information. A CDB Extension should only contain vendor or device specific information. CDB content adhering to the CDB standard should only be found in the CDB versions. Client devices not concerned with a CDB extension should ignore all non-CDB compliant content, without loss of information.

5.1.7 CDB Attributes Metadata

The CDB attributes are listed and described in section 5.7.1.3 CDB Attributes. The metadata for these attributes is stored in \CDB\Metadata\CDB_Attributes.xml and the schema can be found in \CDB\Metadata\Schema\Vector_Attributes.xsd. In essence, the file is the transposition of the text found in section 5.7.1.3 CDB Attributes into a format more appropriate for a computer program.

Its contents are as follows:

```
<Vector_Attributes>
  <Attributes>
    <Attribute>...</Attribute>
    ...
    <Attribute>...</Attribute>
  </Attributes>
  <Units>
    <Unit>...</Unit>
    ...
    <Unit>...</Unit>
  </Units>
  <Scalers>
    <Scaler>...</Scaler>
    ...
    <Scaler>...</Scaler>
  </Scalers>
</Version>
```

The file is composed of three major sections, the first one being the most important. The file has a list of attributes, followed by two lists of units and scalars that are referenced by individual attribute.

5.1.7.1 Definition of the <Attribute> Element

Requirement 75

<http://www.opengis.net/spec/cdb/1.0/core/attribute-definition>

Each attribute *SHALL* be defined as follows:

```
<Attribute code="..." symbol="...">
  <Name>...</Name>
  <Description>...</Description>
  <Level>...</Level>
  <Value>...</Value>
</Attribute>
```

The code is the integer value assigned to each attribute listed in section 5.7.1.3, CDB Attributes. The symbol is the unique character string identifying the attribute. The <Name> is the long form of the symbol. The <Description> is a free form text describing the attribute. The <Level> is defined below and provides the schema level of the attribute. The <Value> element provides the information required to interpret (parse) the value assigned to this attribute.

Requirement 76

<http://www.opengis.net/spec/cdb/1.0/core/attribute-schema-level>

Each schema level *SHALL* be defined as follows:

```
<Level>
  <Instance>...</Instance>
  <Class>...</Class>
  <Extended>...</Extended>
</Level>
```

The <Level> provides a mean to state if the attribute is *Preferred*, *Supported*, *Deprecated*, or *Not Supported* for each of the schema level.

Requirement 77

<http://www.opengis.net/spec/cdb/1.0/core/attribute-value>

Each `<Value>` *SHALL* be defined as follows:

```
<Value>
  <Type>...</Type>
  <Format>...</Format>
  <Precision>...</Precision>
  <Range>...</Range>
  <Unit>...</Unit>
  <Scaler>...</Scaler>
</Value>
```

The `<Type>` is one of *Text*, *Numeric*, or *Boolean*.

In the case of a numeric datatype, the `<Format>` indicates if it is a *Floating-Point* or an *Integer* value. For a floating point type, the `<Precision>` provides the number of digits before and after the decimal point. For numeric types, the `<Range>` provides the minimum and maximum values; the `<Unit>` is a reference to a unit code; and the `<Scaler>` is a reference to a scaler code; both codes being respectively defined in subsequent `<Units>` and `<Scalers>` sections.

5.1.7.2 Definition of the `<Unit>` Element

Requirement 78

<http://www.opengis.net/spec/cdb/1.0/core/attribute-units>

The `<Units>` section *SHALL* be a list of `<Unit>` definitions as follow:

```
<Unit code="..." symbol="...">
  <Name>...</Name>
  <Description>...</Description>
</Unit>
```

The code is a positive integer used as a key when a `<Value>` references a unit. The symbol is the character string that is commonly recognized as the unit identifier. The `<Name>` is the long form of the unit symbol and `<Description>` is a free-form text describing this unit.

5.1.7.3 Definition of the <Scaler> Element

Requirement 79

<http://www.opengis.net/spec/cdb/1.0/core/attribute-scaler>

The <Scalers> section SHALL be a list of <Scaler> definitions as follow:

```
<Scaler code="..." symbol="...">
  <Name>...</Name>
  <Description>...</Description>
  <Multiplier>...</Multiplier>
</Scaler>
```

The code is a positive integer used as a key when a <Value> references a scaler. The symbol is the character string that is commonly recognized as the scaler identifier. The <Name> is the long form of the scaler symbol and <Description> is a free-form text describing this scaler. Finally, <Multiplier> is the numerical multiplier applied to the base unit.

5.1.7.4 Example of CDB_Attributes.xml

The following example illustrates how to define an attribute:

```
<Vector_Attributes>
  <Attributes>
    <Attribute code="2" symbol="A01">
      <Name>Angle of Orientation</Name>
      <Description>Angle of Orientation with greater than 1 degree resolution.
        The angular distance measured from true north (0 deg) clockwise to the
        major (Y) axis of the feature. If the feature is square, the axis 0
        through 89.999 deg shall be recorded. If the feature is circular, 360.000
        deg shall be recorded.
      </Description>
      <Level>
        <Instance>Preferred</Instance>
        <Extended>Supported</Extended>
      </Level>
      <Value>
        <Type>Numeric</Type>
        <Format>Floating-point</Format>
        <Precision>3.3</Precision>
        <Range interval="Right-Open">
          <Min>0</Min>
          <Max>360</Max>
        </Range>
        <Unit>2</Unit>
      </Value>
    </Attribute>
  </Attributes>
</Vector_Attributes>
```

```

<Units>
  <Unit code="2" symbol="deg">
    <Name>degree</Name>
    <Description>To measure an angle</Description>
  </Unit>
</Units>
<Scalers>
  <Scaler code="2" symbol="k">
    <Name>kilo</Name>
    <Description>A multiplier: thousand</Description>
    <Multiplier>1000</Multiplier>
  </Scaler>
</Scalers>
</Vector_Attributes>

```

The schema explains the use of the `interval` attribute of the `<Range>` element.

Requirement 80

<http://www.opengis.net/spec/cdb/1.0/core/ao1>

The angular distance measured from true north (0 deg) clockwise to the major (Y) axis of the feature. If the feature is square, the axis 0 through 89.999 deg *SHALL* be recorded. If the feature is circular, 360.000 deg *SHALL* be recorded.

5.1.8 Geomatics Attributes Metadata

Geomatics attributes (section 5.7.1.2.6.2), are listed in “**Geomatics_Attributes.xml**” (section 3.1.1). The file uses the `Geomatics_Attributes.xsd` schema.

5.1.9 Vendor Attributes Metadata

Vendor attributes (section 5.7.1.2.6.3), are listed in “**Vendor_Attributes.xml**” (section 3.1.1). The file uses the `Vendor_Attributes.xsd` schema.

5.1.10 Configuration Metadata

The Configuration metadata file provides the means of defining CDB Configurations. The syntax of the file is given below. The complete XML schema is provided in `/CDB/Metadata/Schema/Configuration.xsd` delivered with the standard.

```
<Configuration>
  <Comment> An optional comment describing this CDB Configuration. </Comment>
  <Version>
    <Folder path="..."/>
    <Comment> An optional comment describing this CDB Version. </Comment>
    <Specification version="..."/>
    <Extension name="..." version="..."/>
  </Version>
  <!-- Other versions as needed -->
</Configuration>
```

A `<Configuration>` is a list of one or more `<Version>` elements. A `<Version>` has a mandatory `<Folder>` element to provide the path to the CDB Version. The other three (3) elements have the same definitions as that of section 5.1.6, Version Metadata.

5.1.10.1 A Note about Folder Path

The use of a relative path to a CDB Version ensures a greater form of interoperability between operating systems and file systems. However, the CDB standard does not prevent the use of absolute paths³⁰. A relative path is expressed relative to the root of the CDB Version containing the Configuration file.

5.1.10.2 Example

Assume that we want to assemble two CDB Versions into a single CDB Configuration. The first CDB Version is located in `/CDB/myVersion` and has the following Version.xml file.

```
<Version>
  <PreviousIncrementalRootDirectory name="/CDB/theVersion"/>
  <Comment> This is the comment describing myVersion. </Comment>
  <Specification version="3.2"/>
</Version>
```

The second CDB Version complies with version 3.0 of the Specification, is located in `/CDB/theVersion`, and has the following Version.xml file.

³⁰ On Windows, the path can even be specified using the UNC notation.

```

<Version>
  <Comment> This is the comment describing theVersion. </Comment>
</Version>

```

The resulting CDB Configuration is stored in /CDB/myConfiguration and its Configuration.xml file could look like this:

```

<Configuration>
  <Comment>
    This is an example of a CDB Configuration referring to two CDB Versions.
  </Comment>
  <Version>
    <Folder path="../myVersion"/>
    <Comment> This is the comment describing myVersion. </Comment>
    <Specification version="3.2"/>
  </Version>
  <Version>
    <Folder path="../theVersion"/>
    <Comment> This is the comment describing theVersion. </Comment>
    <Specification version="3.0"/>
  </Version>
</Configuration>

```

Notice the use of relative paths to refer to the CDB Versions. Also notice the addition of the <Specification> element to the second <Version> to explicitly state that it contains data complying with version 1.0³¹ of the standard.

5.2 Navigation Library Datasets

The NavData dataset represents the navigation portion of a CDB. It supports several simulation subsystems such as the Instrument Landing System (ILS), Inertial Navigation/Global Positioning System, and Microwave Landing System Communications. The dataset also provides descriptions of airspaces, airways, heliports, helipads, gates, runways, approaches, and terminals. The dataset also provides information regarding climb procedures out of airports.

Requirements Class - Navigation Data (81-84)	
/req/core/nav-data	
Target type	Operations
Dependency	Currently Shapefile format, dBASE

³¹ Was Version 3.0 in the OGC Best Practice.

Dependency	Various XML schema
Requirement 81	/req/core/navigation-file-format
Requirement 82	/req/core/nav-schema-cs2
Requirement 83	/req/core/nav-key-datasets
Requirement 84	/req/core/navdata-component

The NavData dataset is broken down into a collection of 46 (forty-six) components related to the Flight Navigation. Together, these 46 components combine all of the information currently provided by the following two organizations:

- Navigation System DataBase (produced by Jeppesen) around the ARINC Standard 424-16
- Product Standard for the Digital Aeronautical Flight Information File (DAFIF) produced by the National Geospatial Intelligence Agency (NGA)

The component selector 2 is set to 001 for basic navigation records and these files are located in the tiled Navigation dataset directories. The component selector 2 is set to 002 for schema files and a value between 101 and 126 for key datasets. Schema and key datasets are located in the global Navigation directory. Component selector 1 and the file type are as per Table 5-2: Tiled Navigation Dataset. It provides a list of all CDB Navigation components with their designated names and description.

Table 5-2: Component Selectors for Navigation Dataset

CS1	CS2	File Extension	Component Name	Component Description
Dataset 400, NavData				
001-046	002	As appropriate	Schema	Lists the data attributes for the given component
	101-126	As appropriate	Key Dataset	Sorted lists used to perform queries within the NavData

- T002: Schema file
- T101: Storage number search key
- T102: Ident search Key
- T103: ICAO search Key
- T104: Frequency search Key
- T106: IATA search Key
- T107: Type search Key
- T108: Additional Ident 1 search Key

- T109: Additional ICAO 1 search Key
- T110: Channel search Key
- T111: Additional Ident 2 search Key
- T114: Range search Key
- T115: Sequence search Key
- T116: Country search Key
- T117: Boundary search Key
- T118: Code search Key
- T120: Additional Ident 3 search Key
- T121: Reserved
- T122: Additional Type 1 search Key
- T123: Additional ICAO 2 search Key
- T126: Additional Ident 4 search Key

Requirement 81

<http://www.opengis.net/spec/cdb/1.0/core/navigation-file-format>

The Navigation Dataset *SHALL* use a vector data format³². Each of the Navigation features *SHALL* be represented by point features. Each point feature *SHALL* be matched to a group of attributes (described in Volume 12: OGC CDB NavAids Attribution and NavAids Attribution Enumeration Values)³³. Volume 12: OGC CDB NavAids Attribution and NavAids Attribution Enumeration Values specifies the enumeration codes for the Navigation data fields.

5.2.1 Schema Files

The schema file lists the data attributes for the given NavData component. It contains the following columns:

Table 5-4: List of Navigation Schema Attributes

Attribute	Type	Length	Definition
ShortName	String	11	A null-terminated string (ten characters or less). Short-hand name of the attribute used in the tiled vector datasets, (this restriction is for backwards compatibility with the dBASE III+ .dbf format which limits the field names to 10

³² Currently a ShapeFile but in future versions other formats will be specified.

³³ Currently provided by the *.dbf and, optionally, *.dbt portions of the Shapefile format

			characters or less)
DataType	String	255	The data type for the attribute
KeyId	Int	4	Index key for the attribute, used when performing a query. Not all attributes have an assigned index key, as only a few attributes can be used to perform a query. For each attribute with an index key, an index key dataset will be created.

Requirement 82

<http://www.opengis.net/spec/cdb/1.0/core/nav-schema-cs2>

For schema files, the value of CS2 *SHALL* be T002

Each attribute with an index Key (KeyId) has an index key dataset created. The index key dataset includes the last three characters of the KeyId inside the component selector 2 (ex. KeyId 2101 would be dataset component selector 2 – T101).

5.2.1.1 Example

Here is the data content of the schema file for the Airport dataset (D400_S001_T002.dbf):

Table 5-5: Example of a Navigation Schema

ShortName	DataType	KeyId
StoraNumbe	UInt64	2101
AlterNam	String	
AsCoStNumb	UInt64	
BeacoAvail	Bool32	
City	String	
CivMilTyp	CivilMilitaryType	
ClearStatu	ClearanceStatus	
Country	CountryEntry	2116
DayliTim	Float32	
DayTimFram	String	

FlipPage	String	
FuelType	String	
HydElePres	Bool32	
IataCode	String	
IcaoCode	String	2103
Ident	String	2102
IfrCapabil	Bool32	
IslanGrou	String	
Jasu	String	
LonRunLeng	Uint32	
LonRunSurf	PavementType	
MagTruIndi	MagneticTrueIndication	
MagneVaria	Float32	
MgrsPositi	String	
Name	String	
NavIcaCod	String	
NavaiIden	String	
Notam	NotamSystem	
OilType	String	
OperaAgenc	String	
OperaHour	OperatingHours	
Point1	GeoCoordinate	
Remark	String	
ServiRemar	String	
SpeedLimit	Uint32	
SpeLimAlti	Sint32	
StateName	StateEntry	
SupFluTyp	String	
TerraImpac	Bool32	
Timezone	Float32	
TransAltit	Sint32	
TransLeve	Sint32	

As per this example, four Airport attributes can be used to perform queries:

- StoraNumbe (key index 2101)
- Ident (key index 2102)
- IcaoCode (key index 2103)

- Country (key index 2116)

5.2.2 Key Datasets

The index Key Datasets are sorted lists used to perform queries within the NavData.

Requirement 83

<http://www.opengis.net/spec/cdb/1.0/core/nav-key-datasets>

For each attribute that has an index key in the schema file, an index Key Dataset *SHALL* be created. For Key Datasets, the Dataset Component Selector 2 *SHALL* include the last three digits of the index key from the schema file.

Table 5-6: List of Navigation Key Attributes

Attribute	Type	Length	Definition
Value	String	255	Value of the data attribute sorted in increasing order (numbers or characters)
Lat ID	Signed Integer	3	Latitude index of the Geocell which contains the data record
Lon ID	Signed Integer	4	Longitude index of the Geocell which contains the data record
Row ID	Integer	4	Index of the data record in the Geocell starting at 1.

This information can then be used to rapidly lookup which CDB Tile contain the data in the pageable NAV dataset (401) and use the Object ID to access the data record in this dataset.

The Storage number is a Primary Surrogate key that uniquely identifies each record within each NAV dataset sub components.

5.2.2.1 Example

Requirement 84

<http://www.opengis.net/spec/cdb/1.0/core/navdata-component>

For the Airport NavData Component, there *SHALL* be 4 key datasets (for attributes StoraNumber, Ident, IcaoCode and Country):

```
\CDB\Navigation\400_NavData\D400_S001_T101.dbf  
(StoraNumber, key index 2101)
```

```
\CDB\Navigation\400_NavData\D400_S001_T102.dbf  
(Ident, key index 2102)
```

```
\CDB\Navigation\400_NavData\D400_S001_T103.dbf  
(IcaoCode, key index 2103)
```

```
\CDB\Navigation\400_NavData\D400_S001_T116.dbf  
(Country, key index 2116)
```

The following is an excerpt from the D400_S001_T102.dbf file (Key Dataset for the Ident attribute):

Table 5-7: Example of Navigation Keys

Value	Object ID	Lon ID	Lat ID
00CA	2	-117	35
00UT	3	-113	37
00WI	6	-90	44
01LS	4	-92	30
01MT	3	-115	48
01WI	2	-91	44
02P	0	-78	40
03AZ	5	-111	31
03CO	3	-105	40
03GA	5	-84	31
04CA	10	-118	34
04MS	4	-91	32
04NV	1	-116	35
05CL	2	-123	38
05LS	2	-93	31
05UT	0	-111	37

06FA	0	-81	26
06MN	1	-93	47
06MO	7	-95	39
06TE	10	-96	30
07FA	7	-81	25
07MT	1	-107	48

For example, the Airport with Ident 04CA will be found in the Geocell with southwest corner at N34:00:00/W118:00:00. NOTE: For Shapefile implementations this will be the 10th record in the corresponding Shapefile.

Here is an example of the Storage number being used as a reference between Navigation types:

- Type: Approach
- Attributions:
 - StoraNumbe → Storage number (Approach)
 - AirStoNumb → Airport storage number (referenced)

In this case, we see the Approach navigation type referencing the Airport navigation type by using the Airport Storage number.

5.3 CDB Model Textures

The following table provides the Component Selectors associated with all kinds of textures that are usable on geotypical (GT), geospecific (GS), moving (MM), and tiled (T2D) models.

In the context of CDB model textures, the first component selector is known as the “Texture Kind” and the second component selector is simply called the “Texture Index”. Column 1 lists all texture kinds supported by the CDB standard. The second column gives the range of indices allowed for each kind.

Table 5-8: Component Selectors for CDB Model Textures³⁴

CS1 (Kind)	CS2 (Index)	Component Name	Component Description
001	001	Year-Round Texture	Base textures for year-round usage on model shells or general base textures for model interiors.
002	001..012	Monthly Texture	Base textures for monthly usage on the shell of models (enumeration values in Annex O, details in section 6.13.5.2)
003	001..004	Seasonal	Deprecated – Replaced with kind 009

³⁴ Annex O can be found in Volume 2 CDB Core Model and Physical Structure Annexes

		Texture	
004	001..999	Uniform Paint Scheme	Base textures for Moving Models with Uniform Paint Schemes (enumeration values in Annex O, details in section 6.13.5.2)
005	001..999	Camouflage Paint Scheme	Base textures for Moving Models with Camouflage Paint Schemes (enumeration values in Annex O, details in section 6.13.5.2)
006	001..999	Airline Paint Scheme	Base textures for Moving Models with Airline Paint Schemes (enumeration values in Annex O, details in section 6.13.5.2)
007	001..999	Shadow Map	Base textures of Moving Models Shadows to be projected onto terrain and/or culture (details in section 6.13.5.1.2)
008	001..999	Motion Blur Texture	Base textures for use with rotating parts (details in section 6.9.2.3)
009	001..004	Quarterly Texture	Base textures for quarterly usage on the shell of models (enumeration values in Annex O, details in section 6.13.5.2)
051	001..999	Night Map	Subordinate textures to simulate the effect of lights inside 3D model shells (details in section 6.13.5.3)
052	001..999	Tangent-Space Normal Map	Subordinate textures used to simulate the effect of irregular surfaces (details in section 6.13.5.5)
053	001..999	Light Map	Subordinate textures to simulate the effect of lights on surrounding surfaces (detail in section 6.13.5.4)
054	001..999	Contaminant	Subordinate textures to represent the presence of particules on runways, taxiways, and roads in general (enumeration values in Annex O, details in section 6.13.5.7)
055	001..999	Skid Mark	Subordinate textures to represent the visible mark left by any solid which moves against another one; especially marks of tires on roads and runways (enumeration values in Annex O, details in section 6.13.5.7)
056	001..999	Detail Texture	Subordinate texture used to add detail to the surface. In most cases, modelers use detail textures to add a finer scaled texture to the base texture (details in section 6.13.5.6)
057	001..999	Cubic Reflection Map	Subordinate textures to simulate reflective surfaces (details in section 6.13.5.8)
058	001..999	Gloss Map	Subordinate textures providing the glossiness of a surface on a per-pixel basis (details in section 6.13.5.9)
099	001	Night Map	Deprecated – Replaced with kind 051

	002	Bump Map	Deprecated – Replaced with kind 052
	003	Light Map	Deprecated – Replaced with kind 053

Annex O enumerates all textures allocated to kind 002, 003, 004, 005, 006, and 055 (See footnote 32).

5.4 GTModel Library Datasets

Table 5-9 provides the component selector values associated with all GTModel datasets.

Table 5-9: Component Selectors for GTModel Datasets

CS1	CS2	File Extension	Component Name	Component Description
Dataset 500, GTModelGeometry				
001	001	*.flt	Geometry Entry File	Files containing the references to both the shell and interiors of all levels of detail of geotypical models.
Dataset 510, GTModelGeometry				
001	001	*.flt	Geometry Level of Detail	Files containing the geometry of the shell of geotypical models for a given level of detail.
Dataset 506, GTModelInteriorGeometry				
001	001..999	*.flt	Interior Geometry	Files describing the geometry of the interior of geotypical models for a given level of detail. The value of Component Selector 2 is the file number. Multiple files are used when the complexity of the interior justifies using more than one file.
Dataset 503, GTModelDescriptor Dataset 508, GTModelInteriorDescriptor				
001	001	.xml	Descriptor	Provides the metadata associated with a GTModel. See section 6.14, Metadata, for a description of the content. <u>NOTE:</u> A model descriptor includes a Composite Material Table for the exclusive use by its corresponding model geometry datasets above. This CMT is not to be confused with the GTModelCMT and GTModelInteriorCMT datasets below.
Dataset 511, GTModelTexture Dataset 507, GTModelInteriorTexture				
-	-	*.rgb	Texture	Individual base and subordinate textures applied on model geometry, see the complete list in section 5.3, CDB Model Textures.
Dataset 504, GTModelMaterial Dataset 509, GTModelInteriorMaterial				
001	001..255	*.tif	Composite Material	Each texel is an index into the associated Composite Material Table (dataset 505 and 513 below). CS2 is

			Index	the layer number.
002	001..254	*.tif	Composite Material Mixture	Each texel indicates the proportion (between 0.0 and 1.0) of the composite material found in the corresponding material layer. Component Selector 2 is the layer number. When the texels are of integral types, they are scaled to the range 0.0 to 1.0.
Dataset 505, GTModelCMT Dataset 513, GTModelInteriorCMT				
001	001	*.xml	Composite Material Table	The Composite Material Table is associated with Material Textures; it contains the definition of the composite materials referenced by the model material datasets above. Its format is as specified in section 2.5.2.2, Composite Material Tables (CMT)
Dataset 512, GTModelSignature (See notes 1 and 2 below)				
001..999	001..016	*.shp *.shx *.dbf	RCS Signature	The Radar Cross Section of a geotypical mode liss described in Volume 5 of this standard.
	017..032	*.dbf	RCS Class Attributes	The class-level attributes associated with the RCS Signature file as described in Volume 5 of this standard.

Note 1: For GTModelSignature dataset, CS1 refers to the “RCS Frequency” and is used to indicate at which frequency (in MegaHertz) the dataset was generated for. The value of CS1 represents a power of 10 of the frequency and ranges from 1 to 999. The range of frequencies that can be represented is from 10^1 MHz to 10^{999} MHz.

Note 2: For GTModelSignature dataset, CS2 refers to the “RCS Polarization Type” and is used to indicate how the electromagnetic field is polarized at transmission and reception by typical Radar. The value can range from 1 to 16 for the instanced-level attributes and from 17 to 32 for the class-level attributes.

Polarization Type (CS2)		Description
Instance Attribute	Class Attribute	
1	17	LINEAR Polarization
2	18	CIRCULAR Polarization
3	19	ELLIPTICAL Polarization
4	20	SINGLE HH Polarization
5	21	SINGLE HV Polarization
6	22	SINGLE VV Polarization
7	23	SINGLE VH Polarization
8	24	DUAL HH-HV Polarization
9	25	DUAL VV-VH Polarization
10	26	DUAL HH-VV Polarization
11	27	ALTERNATING HH-HV Polarization
12	28	ALTERNATING VV-VH Polarization
13	29	POLARIMETRIC HH Polarization
14	30	POLARIMETRIC VV Polarization
15	31	POLARIMETRIC HV Polarization
16	32	POLARIMETRIC VH Polarization

5.5 MModel Library Datasets

Table 5-10 provides provide the component selector values associated with all Mmodel datasets.

Table 5-10: Component Selectors for Mmodel Datasets

CS1	CS2	File Extension	Component Name	Component Description
Dataset 600, MmodelGeometry (See note 1 below)				
001..999	001..999	*.flt	Geometry	Files containing the geometry of Mmodels as described in Chapter 6.
Dataset 601, MmodelTexture				
-	-	*.rgb	Texture	Individual base and subordinate textures applied on model geometry, see the complete list in section 5.3, CDB Model Textures.
Dataset 603, MmodelDescriptor				
001	001	*.xml	Descriptor	Provides the metadata associated with a Mmodel. See section 6.14, Metadata, for a description of the content. NOTE: The MmodelDescriptor includes a Composite Material Table for the exclusive use by the MmodelGeometry dataset. This CMT is not to be confused with the MModelCMT dataset below.
Dataset 604, MmodelMaterial				
001	001..255	*.tif	Composite Material Index	Each texel is an index into the Composite Material Table (dataset 605). Component selector 2 is the layer number.
002	001..254	*.tif	Composite	Each texel indicates the proportion (between 0.0

			Material Mixture	and 1.0) of the composite material found in the corresponding material layer. Component Selector 2 is the layer number. When the texels are of integral types, they are scaled to the range 0.0 to 1.0.
Dataset 605, MModelCMT				
001	001	*.xml	Composite Material Table	This is the composite material table for use with MmodelMaterial dataset. Its content is described in section 2.5.2.2, Composite Material Tables (CMT).
Dataset 606, MmodelSignature (See notes 2 and 3 below)				
001..999	001..016	*.shp *.shx *.dbf	RCS Signature	The encoding of the Radar Cross Section of a moving model is described in Volume 5 of this standard.
	017..032	*.dbf	RCS Class Attributes	The class-level attributes associated with the RCS Signature file as described in Volume 5 of this standard.

Note 1: For the MmodelGeometry dataset, the geometry of a moving model can be made of one or more parts, each stored in one or more files depending on how complex a part is.

The value of CS1 represents the part number. A Moving Model has at least one part, the model itself that is also used as a master file for all the other parts when applicable. This is part number 1. Other parts are numbered sequentially. An example of an extra part is a removable external fuel tank.

The value of CS2 is the file number. This number is used when the complexity of a part requires using more than one file. The file number starts with 1. NOTE: A part that references external files does it through OpenFlight Xref nodes.

Note 2: For MmodelSignature dataset, CS1 refers to the “RCS Frequency” and is used to indicate the range of frequencies (in MegaHertz) the dataset was generated for. The range is the set of frequencies from 10^{CS1-1} MHz without exceeding 10^{CS1} MHz.

Note 3: For MmodelSignature datasets, CS2 refers to the “RCS Polarization Type” and is used to indicate how the electromagnetic field is polarized at transmission and reception by typical Radar. The value can range from 1 to 16 for the instance-level attributes of polarizations and from 17 to 32 for the class-level attributes of polarizations.

Polarization Type (CS2)		Description
Instance Attribute	Class Attribute	
1	17	LINEAR Polarization
2	18	CIRCULAR Polarization
3	19	ELLIPTICAL Polarization
4	20	SINGLE HH Polarization
5	21	SINGLE HV Polarization
6	22	SINGLE VV Polarization
7	23	SINGLE VH Polarization
8	24	DUAL HH-HV Polarization
9	25	DUAL VV-VH Polarization
10	26	DUAL HH-VV Polarization
11	27	ALTERNATING HH-HV Polarization
12	28	ALTERNATING VV-VH Polarization
13	29	POLARIMETRIC HH Polarization
14	30	POLARIMETRIC VV Polarization
15	31	POLARIMETRIC HV Polarization
16	32	POLARIMETRIC VH Polarization

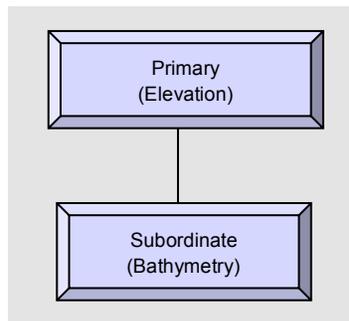
5.6 Tiled Raster Datasets

A raster dataset consist in an evenly spaced grid of data elements that are positioned (in geographic units) along the north-south and east-west axis. This section describes all of the CDB raster datasets.

Requirements Class - Tiled Raster General (85-87)	
/req/core/tiled-raster-datasets-general	
Target type	Operations
Dependency	Various XML schema
Requirement 85	/req/core/tile-grid-structure
Requirement 86	/req/core/tile-implicit-corner-computation
Requirement 87	/req/core/tile-implicit-center-computation

Most of the CDB raster datasets are broken down further into components. A component is a specialization of the dataset. For example, bathymetry is a specialization of altimetry data because it is targeted to the representation of submerged terrain surfaces; the bathymetric depth data represents altimetry (e.g., heights) with respect to the Primary Elevation component. Together, the Primary Elevation and Bathymetry components form the Elevation Dataset.

A component can be either 1) “primary”, (i.e., it can be used on a stand-alone basis) or 2) “subordinate”, (i.e., it must be used in conjunction with one or more primary components and one or more subordinate components). Subordinate components depend on information contained within another component. Subordinate components are used to progressively add complexity and/or information to a primary component or to another subordinate component. For instance, the Elevation component is a primary component that contains information to allow a simulator client-device to accurately represent the terrain profile or determine the terrain height. On the other hand, Bathymetry is a subordinate component because it cannot be used stand-alone and that it is implicitly subordinate to the Elevation component. It uses the Elevation component to determine the depth of an ocean.

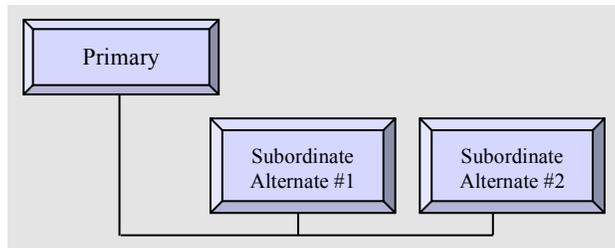


In addition to the notion of primary and subordinate, the CDB embodies the notion of Component Alternates. A component is said to be an alternate component if it can be used interchangeably with other components.

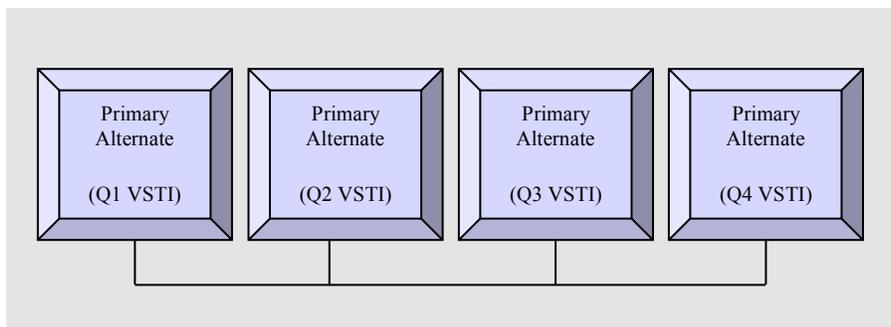
The two concepts can be used in combination as follows:

- Primary
- Subordinate or
- Primary Alternate
- Subordinate Alternate

A component is Primary if the component is not dependent on another component, i.e., it can be used on a stand-alone basis. Conversely, a component is said to be Subordinate if the dataset is dependent on another component, be it a primary component or another subordinate component. For example, the Bathymetric component is referenced to the CDB Primary terrain elevation component; as a result, it is a subordinate component. The Primary elevation component is a primary component because it does not depend on any other component.



A component is Primary Alternate if a) the component is not dependent on another component, be it a primary component or another subordinate component and b) other primary components can be used interchangeably with the component. For example, the VSTI Q1, Q2, Q3 and Q4 components are all primary alternate components, because they each form the primary layer of terrain imagery, yet they can be used interchangeably.



Finally, a component is Subordinate Alternate if a) if the component is dependent on another component, be it a primary component or another subordinate component and b) other subordinate components can be used interchangeably with the component.

In the case of alternate components, one of them is designated as the default component; in the event that an alternate component is missing in the CDB, the client-devices are required to revert to the default alternate component.

Since subordinate components usually improve the overall fidelity of the dataset, client-devices can revert to the primary component in the event that a subordinate component is missing in the CDB. This behavior allows the CDB standard to meet one important objective which is to allow any simulator client-device with relatively low performance to still be able to run a CDB implementation (scalability).

Conceptually, the raster dataset tile's internal grid structure uniformly subdivides the tile in both axes. The main characteristic of raster tile is that the number of data elements and the position of every data element are implicit. When processing raster data, the application should derive the data element position from the geodetic tile position.

Requirement 85

<http://www.opengis.net/spec/cdb/1.0/core/tile-grid-structure>

The tile grid structure *SHALL* be aligned to the tile edge boundary. The grid elements *SHALL* be organized in row, column order, starting from the northwest corner scanning towards the southeast. This is true for tiles in both the south and north hemisphere.

The CDB standard accommodates for data elements that can be aligned either to the centers or to the corners of the internal tile grid structure. In both cases, the number of data elements in the tile is a power of two. Furthermore, data elements can either represent values representative of samples on the earth surface (e.g., altitude at a point) or values representative of a surface area on the earth surface (average altitude over square area).

Figure 5-1: Center Grid Data Elements, illustrates a CDB tile with a grid of “center grid data elements” overlaid onto the tile’s grid structure with the addressing conventions and the alignment of the samples and areas assigned to each of the data elements.

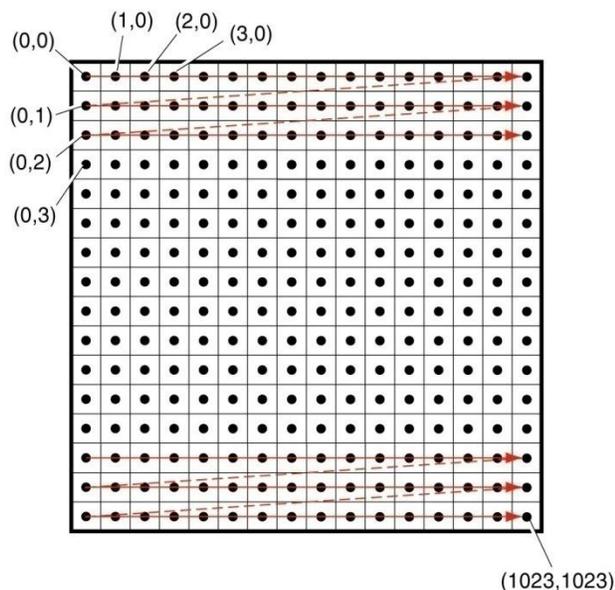


Figure 5- 1: Center Grid Data Elements

Figure 5-2: Corner Grid Data Elements, illustrates a CDB tile with a grid of “corner grid data elements” overlaid onto the tile’s grid structure with the addressing conventions and the alignment of the samples and areas corresponding to the data elements.

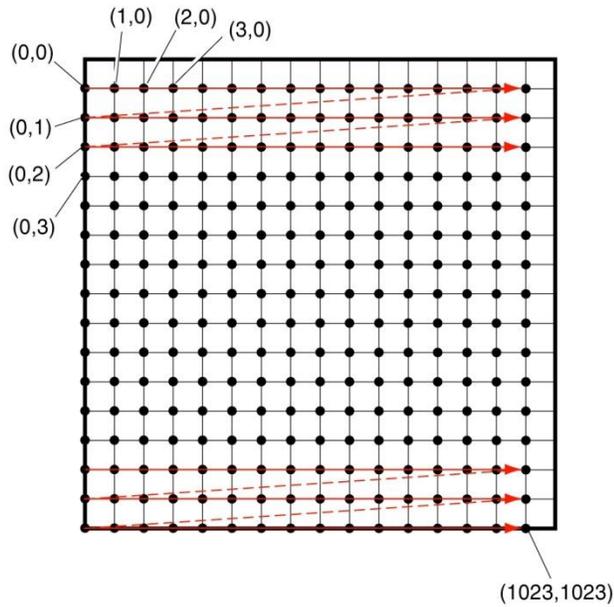


Figure 5- 2: Corner Grid Elements

Figure 5-3: Center Data Elements as a Function of LODs, illustrates an implementation of center grid data elements with four levels-of-details. Note the shift in position of the data element centers along the x- and y-axis as we shift to progressively coarser levels-of-detail. Note also that the edges of the data elements areas stay aligned with x- and y-axis as we shift to progressively coarser levels-of-detail.

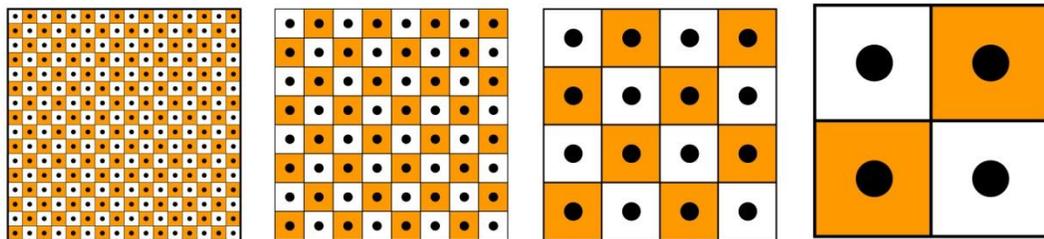


Figure 5- 3: Center Data Elements as a Function of LODs

Figure 5-4: Corner Data Elements as a Function of LODs, illustrates an implementation of corner grid data elements with four levels-of-details. Note the shift in the edge of the data element area along the x- and y-axis as we shift to progressively coarser levels-of-detail. Note also that the position of the data elements areas stay aligned with x- and y-axis as we shift to progressively coarser levels-of-detail.

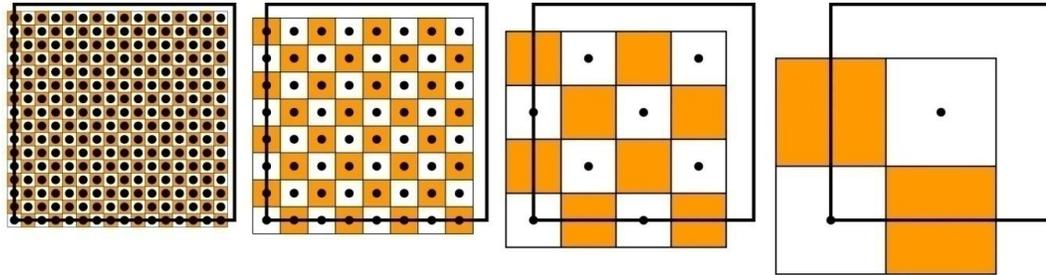


Figure 5-4: Corner Data Elements as a Function of LODs

Sections 5.6.1, 5.6.2, and 5.6.3 describe all of the raster datasets, namely the Tiled Elevation Dataset, the Tiled Imagery Dataset, and the Tiled Raster Material Dataset; each of these sections describes the associated “corner” versus “center” conventions. This convention is intrinsic to the corresponding dataset and is not parametrical. Any changes to these implicit properties require an additional specific dataset to ensure compatibility with applications.

Requirement 86

<http://www.opengis.net/spec/cdb/1.0/core/tile-implicit-corner-computation>

The latitude and longitude of an implicit corner data element (in a tile) with coordinates (i, j) SHALL be computed as per the following equation (note the equation for $Xunit_{LOD}$ and $Yunit_{LOD}$ can be found in eq. 3-4)

$$\begin{aligned} Latitude &= TileLatitude - (j \times YUnit_{LOD}) \\ Longitude &= TileLongitude + (i \times XUnit_{LOD}) \end{aligned} \tag{eq. 5-1}$$

Where $TileLatitude$, $TileLongitude$ are the tile coordinates at the upper left corner and (i, j) are the data element coordinates.

Requirement 87

<http://www.opengis.net/spec/cdb/1.0/core/tile-implicit-center-computation>

Similarly, the position of an implicit center data element in a tile *SHALL* be computed as per the following equation (5-2):

$$\begin{aligned} \text{Latitude} &= \text{TileLatitude} - ((j + 0.5) \times YUnit_{LOD}) \\ \text{Longitude} &= \text{TileLongitude} + ((i + 0.5) \times XUnit_{LOD}) \end{aligned}$$

5.6.1 Tiled Elevation Dataset

In a CDB, terrain elevation is depicted by a grid of data elements at regular geographic intervals and at prescribed locations within the tile; each grid element is associated with an elevation value. The resultant is a Digital Elevation Model (DEM) of the earth surface with respect (above or below) to the WGS-84 reference ellipsoid. The Elevation Dataset implicitly follows the corner grid element conventions.

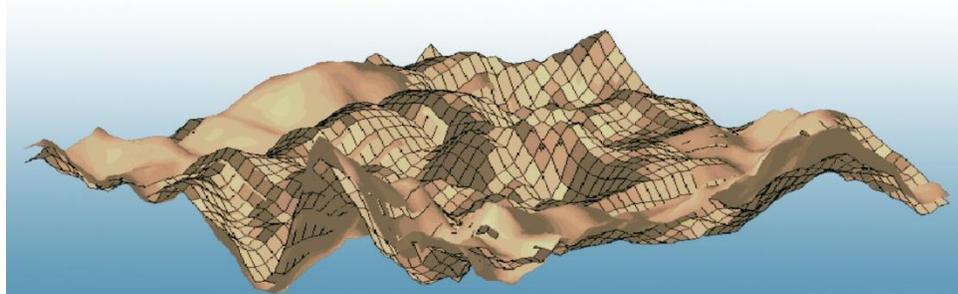


Figure 5- 5: Example of Digital Elevation Model (DEM)

The (x, y) coordinates of each grid element are its longitude and latitude, respectively. The Elevation dataset holds the vertical extent of the terrain. In Figure 5-6: DEM Depicted as a Grid of Elevations at Regular Sample Points, obstacles such as a tower and a building have been overlaid on the terrain grid to demonstrate that obstacle heights are relative to the terrain height.

Requirements Class - Tiled Raster Elevation/Terrain (88-95)	
/req/core/tiled-raster-elevation-terrain	
Target type	Operations

Dependency	Various XML schema
Requirement 88	/req/core/primary-terrain-component
Requirement 89	/req/core/terrain-tiff-channel1
Requirement 90	/req/core/terrain-tiff-channel2
Requirement 91	/req/core/terrain-hypsography-offline
Requirement 92	/req/core/terrain-online-terrain-constraints
Requirement 93	/req/core/terrain-lpn-use
Requirement 94	/req/core/min-max-elevation-data-type
Requirement 95	/req/core/maxculture-data-type

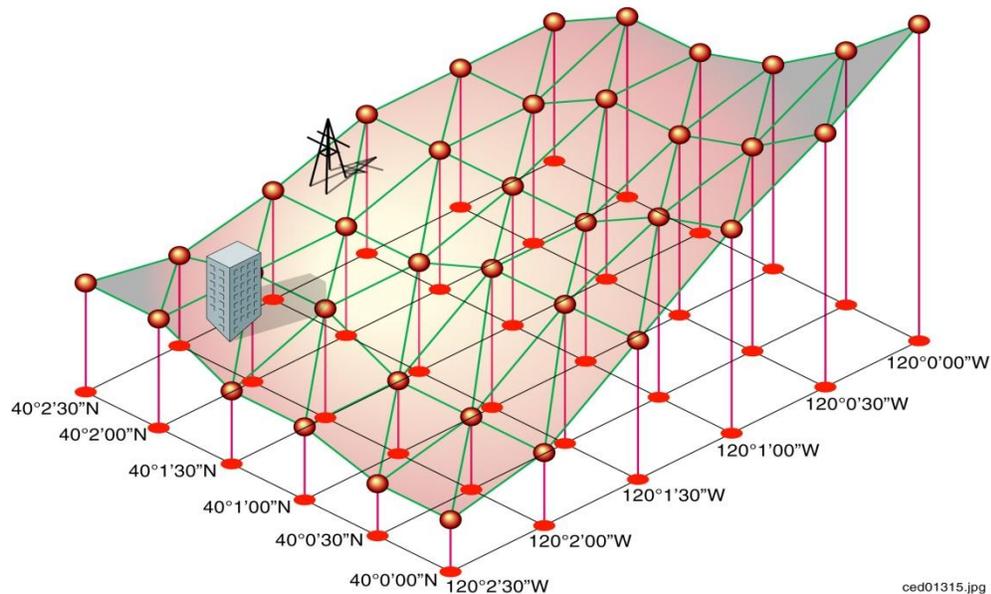


Figure 5- 6: DEM Depicted as a Grid of Elevations at Regular Sample Points

The CDB LOD structure lends itself to variable levels of terrain elevation fidelity, on a per Tile-LOD basis. The selected grid spacing is a function of the height and geographic precision that is desired. Through the use of LODs, one can specify a grid spacing appropriate to the required terrain fidelity requirements. For instance, the accurate depiction of a runway profile (say down to 1 ft height precision) would typically require a relatively fine pitch terrain elevation LOD even if the area is nominally flat. Similarly, the accurate representation of sharp altitude discontinuities (e.g., cliffs) also requires increasingly finer elevation grids to capture the cliff profile correctly.

Negative elevation values do not imply that the elevation point is submerged; rather, a negative value merely indicates that its altitude is below the WGS-84 reference ellipsoid.

The CDB standard defines a number of subordinate elevation components that are used in combination with the primary component of the Elevation Dataset.

5.6.1.1 Terrain Mesh Types

The CDB standard defines two mesh types to connect each grid post to its neighbors. The purpose of the mesh type is to minimize the error in the representation of the Terrain Profile built from the components of the Elevation dataset. Figure 5-7 below illustrates the supported CDB Mesh Types.

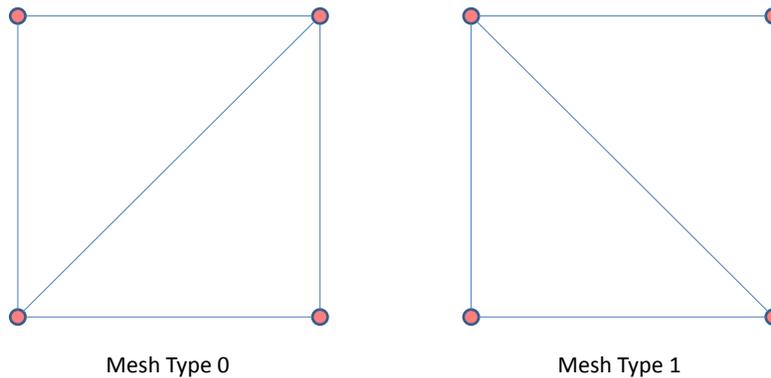


Figure 5-7: CDB Mesh Types

Mesh type 0 connects the southwest grid post to its northeast neighbor while mesh type 1 does the same for the northwest and southeast posts.

5.6.1.1.1 Data Type

The mesh type is represented by an unsigned integer of a size that is large enough to accommodate the range of mesh types. Currently, there are only two values defined; as such, an 8-bit unsigned integer is sufficient and appropriate to store the mesh type.

5.6.1.1.2 Default Value

By default, when the mesh type is not specified or not available, a value of zero is assumed.

5.6.1.2 List of all Elevation Dataset Components

The Elevation Dataset is comprised of several components listed here and detailed in the subsequent sections.

Table 5-11: Elevation Dataset Components

CS1	CS2	File Extension	Component Name	Component Description
Dataset 001, Elevation				
001	001	*.tif	Primary Terrain Elevation	A grid of data representing the Elevation at the surface of the Earth. Stored as a 1 or 2-channel TIFF image. When present, the second channel provides the mesh type of each grid element.
001	002	*.tif	Primary Terrain Elevation Control	Deprecated
001	003	*.tif	Primary Alternate Terrain Elevation	A grid of data representing the Elevation of the surface of the Earth at specified Latitude and Longitude offsets inside each grid element. Stored as a 4-channel TIFF image.
002..099	001	*.tif	Subordinate Terrain Elevation	Deprecated
002..099	002	*.tif	Subordinate Terrain Elevation Control	Deprecated
100	001	*.tif	Subordinate Bathymetry	A grid of data representing the Depth of water with respect to the selected Terrain Elevation component. Store as a 1 or 2-channel TIFF image. When present, the second channel provides the mesh type of each grid element.
100	002	*.tif	Subordinate Alternate Bathymetry	A grid of data representing the Depth of water at specified Latitude and Longitude offsets inside each grid element with respect to the selected Terrain Elevation component. Stored as a 4-channel TIFF image.
101	001	*.tif	Subordinate Tide Elevation	A grid of data representing the average height variation of water with respect to the Primary Terrain Elevation Component.
Dataset 002, MinMaxElevation				
001	001	*.tif	Minimum Elevation	Minimum height (on a per tile LOD basis) of the Primary Terrain Elevation Dataset Component (excluding all cultural features).
001	002	*.tif	Maximum Elevation	Maximum height (on a per tile LOD basis) of the Primary Terrain Elevation Dataset Component (excluding all cultural features).

Dataset 003, MaxCulture				
001	001	*.tif	Primary Maximum Culture Elevation	Maximum height (on a per tile LOD basis) of the bounding boxes of all cultural features held in the vector tiled datasets within the geographic footprint of the area represented by the sample value.

5.6.1.3 Primary Terrain Elevation Component

The Primary Terrain Elevation component of the Elevation dataset represents the surface of the Earth, i.e., the emerged part of the Earth's crust, the surface of persistent bodies of water (e.g., ocean, lakes, rivers), and the permanent ice-covered parts of the Earth. However, the Primary Terrain Elevation values exclude the heights of natural vegetation and man-made cultural features.

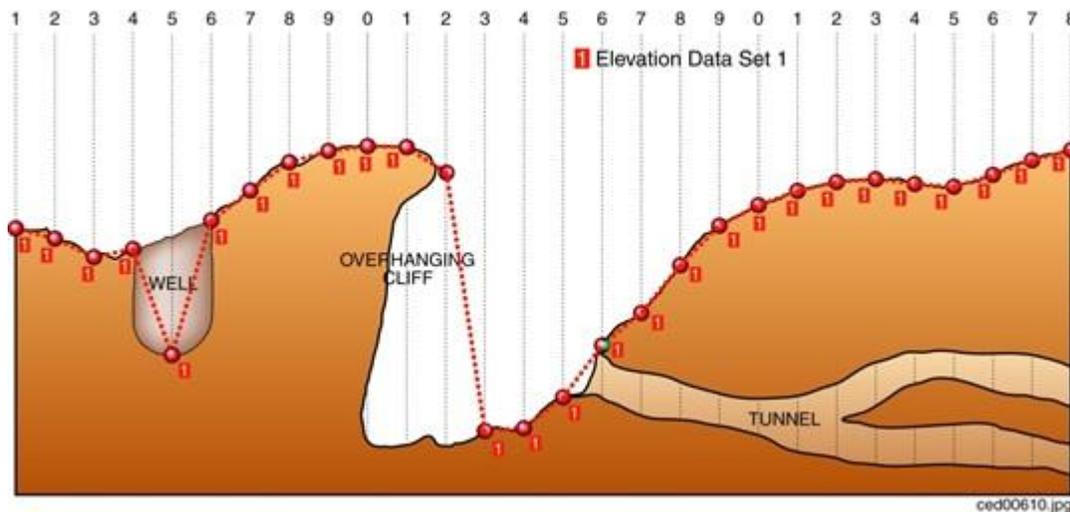


Figure 5-8: Primary Terrain Elevation Component

By definition, the Primary Terrain Elevation component represents a single elevation value at each grid element of the dataset. As a result, each value of the Primary Terrain Elevation component corresponds to the elevation of the highest Earth surface at the specified latitude and longitude coordinate. Consider the example illustrated in Figure 5-8: Primary Terrain Elevation Component. The diagram illustrates a region of Earth with a well, an overhanging cliff, and a network of tunnels. Using solely the Primary Terrain Elevation component, the resulting terrain representation corresponds to the continuous terrain profile represented by the red dotted line; as a result, the underside of the overhanging cliff, the tunnels, and the vertical walls of the well are not represented.

To represent terrain walls, overhanging cliffs, wells, tunnels and mineshafts, modelers are required to supplement the Primary Terrain Elevation component with terrain-conformed 3D models as illustrated in Figure 5-9: Modeling of Wells, Overhanging Cliffs and Tunnels.

Embedded within such 3D models are special cutout zones which represent the clipping geometry that is used to cut out the terrain skin.

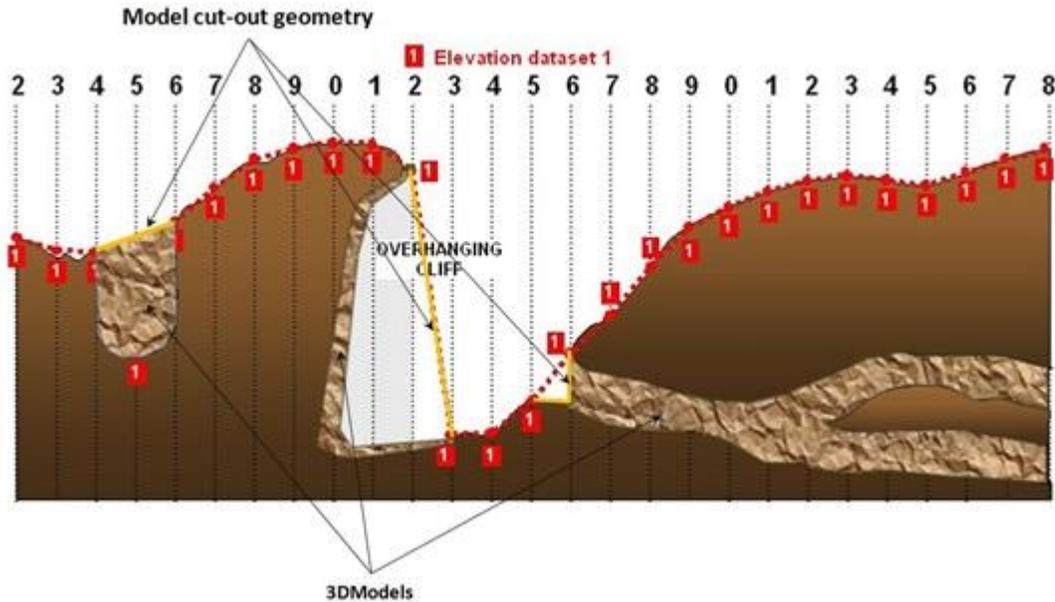


Figure 5-9: Modeling of Wells, Overhanging Cliffs and Tunnels

Model cutouts are explained in section 6.5.6.3, Model Cutout Zones and model conforming modes are described in section 6.7, Model Conforming.

5.6.1.3.1 Data Type

Requirement 88

<http://www.opengis.net/spec/cdb/1.0/core/primary-terrain-component>

The Primary Terrain Elevation component of the Elevation dataset *SHALL* be represented as a 1 or 2-channel TIFF image.

The first channel contains the *Elevation* of the grid post; the optional second channel indicates the *Mesh Type* used to connect the four grid posts that are adjacent to the grid element. The elevation is represented by a floating-point or signed integer value expressed in meters and relative to the WGS-84 reference ellipsoid. Integer values for tiles at LOD larger than 0 are scaled according to the following formula:

$$\text{Elevation}=\text{IntValue}\times 2\text{-LOD}$$

Integer values can make use of TIFF's 8-bit, 16-bit, or 32-bit representation.

The *Mesh Type* is stored as an unsigned 8-bit integer.

5.6.1.3.2 Default Read Value

Simulator client-devices should assume an *Elevation* value of *Default Elevation-1* if the data values of the Primary Terrain Elevation component are not available (files associated with the Primary Terrain Elevation component for the area covered by a tile, at a given LOD or coarser, are either missing or cannot be accessed). The default value is stored in \CDB\Metadata\Defaults.xml. In absence of a default value, the CDB standard states that client-devices use a value of zero.

If the TIFF file has a single channel image, client devices assume a *Mesh Type* of zero.

5.6.1.3.3 Default Write Value

The files associated with the Primary Terrain Elevation component for area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted. If the tool generating the Primary Terrain Elevation does not support the optional Mesh Type, the optional second channel of the file need not be created; in which case the TIFF file becomes a single channel image.

5.6.1.3.4 Supported Compression Algorithm

The CDB standard supports the LZW compression algorithm for the Primary Terrain Elevation component. Consider compressing the file if its content is not of type floating-point.

5.6.1.4 Primary Alternate Terrain Elevation Component

The accurate delineation of man-made elevation features such as roads, railroads, runways, and natural elevation features such as ridgelines, coastlines requires very high levels-of-detail for the Primary Terrain Elevation component. Such cases typically require an elevation grid pitch of approximately ½ m or better (the equivalent of 8 million triangles per square kilometer) resulting in unnecessary large storage and runtime processing. The Primary Alternate Terrain Elevation component offers an effective solution to handle these use-cases.

The Primary Alternate Terrain Elevation component provides the means to accurately delineate terrain features without having to revert to very fine LODs of the Primary Terrain Elevation component. To do this, the Primary Alternate Terrain Elevation component encodes information that re-positions each elevation sample anywhere within its assigned grid element. In other words, the “phase” of each terrain elevation sample can be specified along the latitude and longitude axes. In effect, the Primary Alternate Terrain Elevation component provides the means to locally increase the altimetric precision of the modeled representation of a terrain profile. While it would be possible for a modeler to manually control the position of individual

elevation points, it is expected that the SE tools automate this process by considering elevation constraint points, lineals and polygons (polygons) provided by the modeler.

The constituents of the Primary Alternate Terrain Elevation are the elevation and mesh type at the specified latitude and longitude offsets inside each grid element. These four constituents are represented as 4 channels of a TIFF image.

The latitude and longitude offsets are expressed as unsigned 8-bit integer values that provide position offsets expressed as 1/256th of the grid spacing for the LOD in question. Note that since the movement of each elevation point is constrained to the inside of its respective grid element, it is impossible to disrupt the (regular grid) topology of the elevation grid; furthermore, it is impossible to have elevation points that move outside of the confines of the tile-LOD. Figure 5-10 illustrates the valid offset values inside each grid element of a tile-LOD.

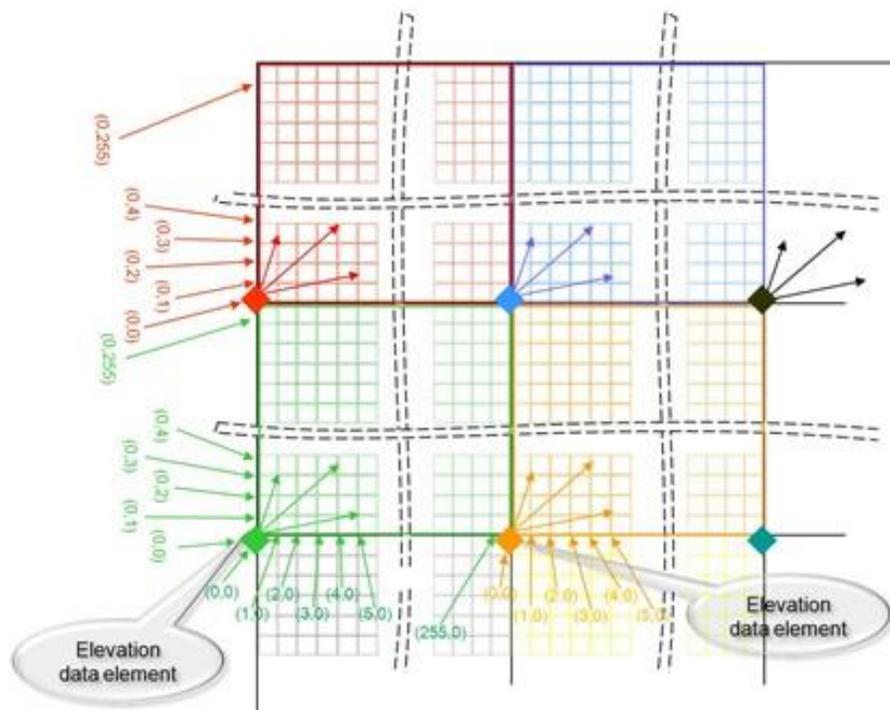


Figure 5-10: Encoding of Lat/Long Offsets

Figure 5-11 illustrates the coverage of grid elements inside a CDB tile. It shows that a grid post is allowed to move inside the area covered by its grid element.

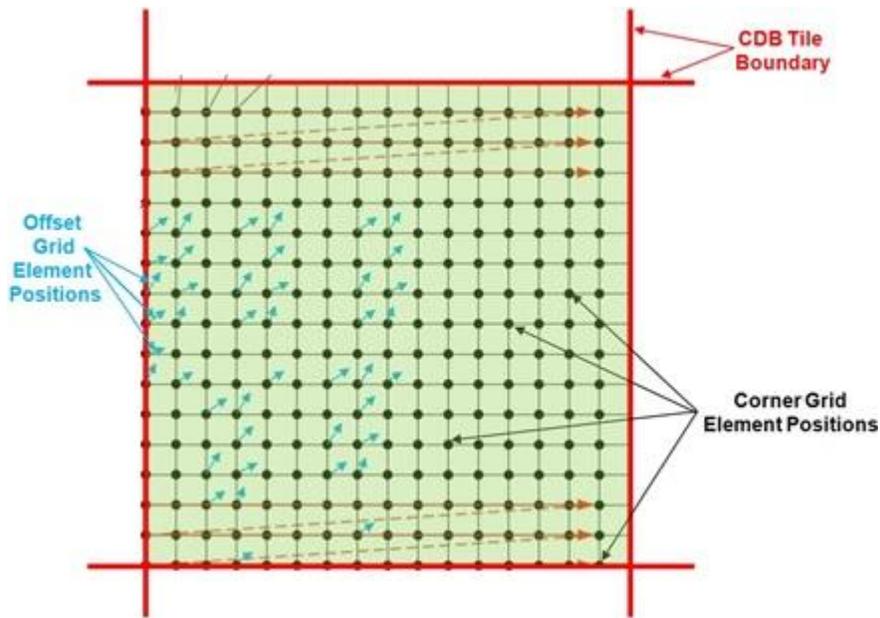


Figure 5-11: Grid Element Coverage within a CDB Tile

The **Latitude Offset** is expressed as an 8-bit unsigned integer value ranging from 0 to 255. The value is scaled so that each grid element is fragmented in 256 equal parts in the latitude direction. Thus, the elevation point cannot be positioned on the latitude of the next grid element directly north of the current grid element.

The **Longitude Offset** is expressed as an 8-bit unsigned integer value ranging from 0 to 255. The value is scaled so that each grid element is fragmented in 256 equal parts in the longitude direction. Thus, the elevation point cannot be positioned on the longitude of the next grid element directly east of the current grid element.

5.6.1.4.1 Data Type

Requirement 89

<http://www.opengis.net/spec/cdb/1.0/core/terrain-tiff-channel1>

The first channel of the TIFF image *SHALL* contain the **Elevation** component and is represented as a floating-point or signed integer value. Integer values for tiles at LOD larger than 0 *SHALL* be scaled according to the following formula:

$$Elevation = IntValue \times 2^{-LOD}$$

Integer values can make use of TIFF's 8-bit, 16-bit, or 32-bit representation.

Requirement 90

<http://www.opengis.net/spec/cdb/1.0/core/terrain-tiff-channel2>

The second channel of the TIFF image contains the *Mesh Type* and *SHALL* be stored as an unsigned 8-bit integer.

The third and fourth channels contain the *Latitude* and *Longitude Offset* components and *SHALL* be stored as unsigned 8-bit integers.

5.6.1.4.2 Default Read Value

Simulator client-devices should assume an *Elevation* value of *Default_Elevation-1* if the data values of the Primary Alternate Terrain Elevation component are not available (files associated with the Primary Alternate Terrain Elevation component for the area covered by a tile, at a given LOD or coarser, are either missing or cannot be accessed). The default value is stored in \CDB\Metadata\Defaults.xml. In absence of a default value, the CDB Specification states that client-devices use a value of zero.

The default *Mesh Type*, *Latitude* and *Longitude Offsets* are zero.

5.6.1.4.3 Default Write Value

The files associated with the Primary Alternate Terrain Elevation component for an area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.

5.6.1.4.4 Supported Compression Algorithm

The CDB standard supports the LZW compression algorithm for the Primary Alternate Terrain Elevation component. Consider compressing the file if its content is not of type floating-point.

5.6.1.5 Terrain Constraints

There are many instances where modelers may wish to take advantage of the availability of position and altitude of cultural features in order to locally control the terrain elevation data at a point, along a specified contour line or within a given area. This operation is usually performed off-line by the modeler and requires that the Elevation dataset be edited and re-generated offline.

Table 5-12: Partial List of Hypsography Feature Codes (for Offline Terrain Constraining)

FACC					FACC-FSC Label	FACC Concept Definition
Level 1	Level 2	Level 3	FSC	FACC-FSC		
C					Hypsography	
	A				Relief	
		010	000	CA010-000	Contour_Line	A line connecting points having the same vertical datum value.
		020	000	CA020-000	Ridge_Line	A line representation of a ridge top.
		025	000	CA025-000	Valley_Line	A line representation of the lowest part of a valley.
		026	000	CA026-000	Breakline	Line representing the demarcation of a sudden and significant change in the gradient of the terrain relief.
		030	000	CA030-000	Spot_Elevation	A designated location with an elevation value relative to a vertical datum.
		035	000	CA035-000	Water_Elevation	A location with a generalized elevation value relative to a vertical datum associated with an inland, usually confined, water body.
		040	000	CA040-000	Contour_Polygon	An arbitrary area outline created to establish elevation as polygons.
		050	000	CA050-000	Surface_Triangle	A set of three spot elevations defining a terrain region of constant slope and aspect.

The Data Dictionary of CDB standard makes provision for the representation of many hypsography features within the Geopolitical Dataset (see Table 5-12: Partial List of Hypsography Feature Codes (for Offline Terrain Constraining)). By virtue of their semantics, these features have no associated modeled representation. The modeler can use these hypsography features to control the generation of the Terrain Elevation grid during the off-line CDB compilation process. This terrain constraining operation can be performed by the SE tools as the CDB is “assembled and compiled”. Note that runtime client-devices do not constrain the Terrain Elevation Dataset to hypsography features.

Requirement 91

<http://www.opengis.net/spec/cdb/1.0/core/terrain-hypsography-offline>

When terrain constraining is performed off-line, hypsography features *SHALL* have AHGT set to True, thereby instructing the SE Tools to constrain the terrain elevation

using the supplied (lat-long-elev) coordinates. The vector feature, currently stored in a vector format, *SHALL* be a PointZ, a MultiPointZ, a PolyLineZ, a PolygonZ or a MultiPatch.

While these hypsography features can be used by the off-line SE tools to control the terrain skinning process, these features can be instead converted into Constraint Features, thereby deferring the terrain constraining process to runtime client-devices.

Constraint Features are features that instruct client-devices to runtime-constrain the terrain Elevation Dataset to a set of prescribed elevation values. This provides modelers the ability to accurately control terrain elevation profiles even if the Terrain Elevation Dataset is of modest resolution and is regularly-gridded; furthermore, the original Elevation Dataset remains unaffected. In effect, the Constraint Features provides a storage-efficient means of capturing terrain contours without having to re-generate / reskin the terrain to a higher-resolution.

Note that this operation is performed on Elevation Datasets that are regularly-gridded or irregularly-gridded. This capability is particularly effective when modelers wish to accurately control terrain elevation profiles but only have regularly-gridded source elevation data of modest resolution at their disposal. Each of these features is associated with vertices that define elevation at the supplied lat-long coordinate(s). This approach provides a level-of-control similar to that of Terrain Irregular Networks (TINs).

The following Constraint Features are used for Online Terrain Constraining:

Requirement 92

<http://www.opengis.net/spec/cdb/1.0/core/terrain-online-terrain-constraints>

For the following constraints, the Feature's AHGT attribute *SHALL* be set to TRUE.

- **Elevation Constraint Point (CA099-000):** In the case of PointZ and MultiPointZ features, the coordinates are used by the client-device to control the terrain elevation at the specified (lat-long). Please note features are ignored that are implemented as Point, PointM, MultiPoint, or MultiPointM types.
- **Elevation Constraint Line (CA099-001):** In the case of PolyLineZ features, the coordinates are used by the client-device to control the terrain elevation at the specified (lat-long). Please note that features are ignored that are implemented as a PolyLine and PolyLineM.
- **Elevation Constraint Area (CA099-002):** In the case of PolygonZ and

MultiPatch features, the coordinates are used by the client-device to control the terrain elevation at the specified (lat-long). Please note features are ignored that are implemented as a Polygon and PolygonM.

Table 5-13: List of Hypsography Feature Codes (for Online Terrain Constraining)

FACC					FACC-FSC Label	FACC Concept Definition
Level 1	Level 2	Level 3	FSC	FACC-FSC		
		099	000	CA099-000	Terrain_Constraint_Point	A point used to constrain the terrain elevation
		099	001	CA099-001	Terrain_Constraint_Line	A line consisting of multiple segments whose vertices and edges are used to constrain the terrain elevation
		099	002	CA099-002	Terrain_Constraint_Area	A multi-triangle convex or concave area whose vertices and edges are used to constrain the terrain elevation

An example of a point-feature is illustrated in Figure 5-12. This picture shows a storage tank located atop a hill. Given the high terrain relief in this area, the modeler is concerned that the terrain may slope significantly in the immediate vicinity of the storage tank, particularly at coarser LODs of the uniform-sampled terrain elevation grid. As a result, he defines a PointZ Constraint Point feature that coincides with the position of the storage tank. AHGT set to True so that the client-device will constrain the Terrain Elevation dataset to the supplied value.



Figure 5-12: Storage Tank Point-Feature

A second example of this principle illustrated in Figure 5-13, this time applied to a road lineal-feature. This picture shows a divided highway running alongside a mountainous area. Given the high terrain relief in this area, the modeler is concerned that the terrain may slope significantly in the immediate vicinity of the road, particularly at the coarser LODs of the uniform-sampled terrain elevation grid. As a result, he defines a PolyLineZ Constraint Lineal feature that coincides with the centerline of the road; AHGT set to True so that the client-device will constrain the Terrain Elevation dataset to the supplied coordinates of the lineal feature.

The CDB standard has well over 50 feature codes whose semantics are related to abstract elevation-related features (such as CA010 Contour line; CA020 Ridge line; CA025 Valley line; CA026 Breakline ...) With the exception of VG018, all of them have semantics that imply a single elevation value. The Feature “Variable Displacement Line”, feature code VG018, is an exception; it allows for a (relative) elevation value for each of the vertices of the “Variable Displacement Line”.

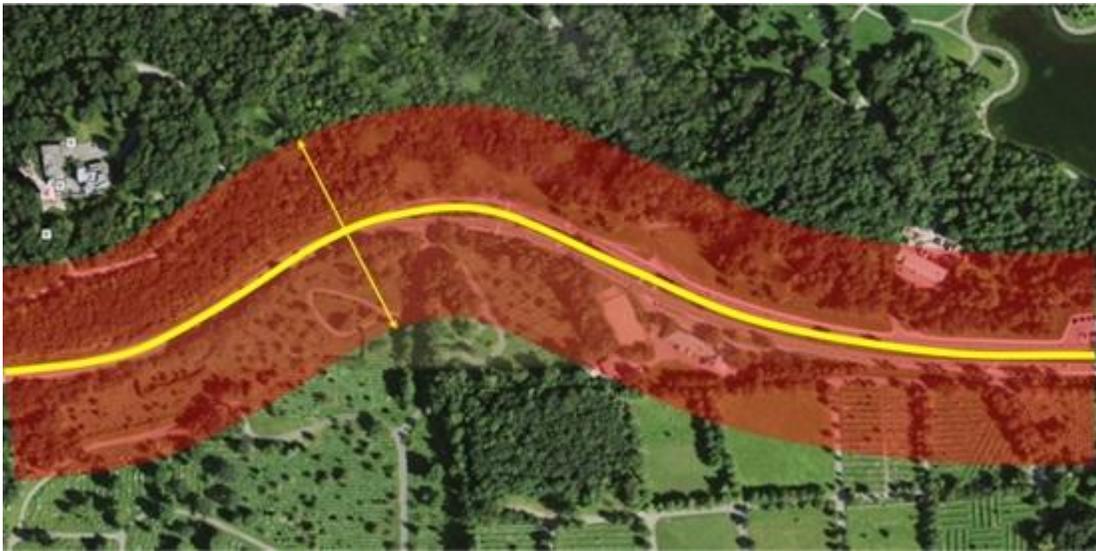


Figure 5-13: Road Lineal Feature

Requirement 93

<http://www.opengis.net/spec/cdb/1.0/core/terrain-lpn-use>

In the case where features overlap one other, client-devices *SHALL* use the Layer Priority Number (LPN) attribute; this attribute is mandatory for geographically overlapping

features.

The LPN attribute is a number in the 0-32767; low numeric values correspond to low priority. The LPN attribute is used to control the order in which the features are applied to (e.g. rendered into) the Elevation dataset. Features are applied in succession in low-to-high priority order into the Terrain Elevation dataset.

5.6.1.6 MinElevation and MaxElevation Components

The MinElevation and MaxElevation components are part of the MinMaxElevation dataset whose purpose is to provide a CDB conformant data store with the necessary data and structure to achieve a high level of determinism in computing line-of-sight intersections with the terrain. The values of each component are with respect to WGS-84 reference ellipsoid. Since both the MinElevation and the MaxElevation values are provided by this standard, any line-of-sight algorithm can rapidly assess an intersection status of the line-of-sight vector with the terrain. An overview of the algorithm governing the line-of-sight computations can be found in Section 8 of Volume 0: OGC CDB Primer (Formerly in Appendix A).

The MinElevation and MaxElevation values follow the “center grid data element” convention of the CDB standard.

The generation of the MinMaxElevation dataset is quite simple. In essence, each center grid element in the MinElevation component represents the lowest altitude for the area represented by that grid element. Likewise, each center grid element in the MaxElevation component represents the highest altitude for the area represented by that grid element.

The MinMaxElevation dataset components are derived from the Primary Terrain Elevation and Primary Alternate Terrain Elevation components. As a result, the MinMaxElevation dataset cannot have more LODs than the Terrain Elevation component it is based on.

5.6.1.6.1 *Level of Details*

As can be seen in Figure 5-14: LOD Structure of Raster Datasets, the MinMaxElevation dataset LODs share the same structure as the Elevation dataset.

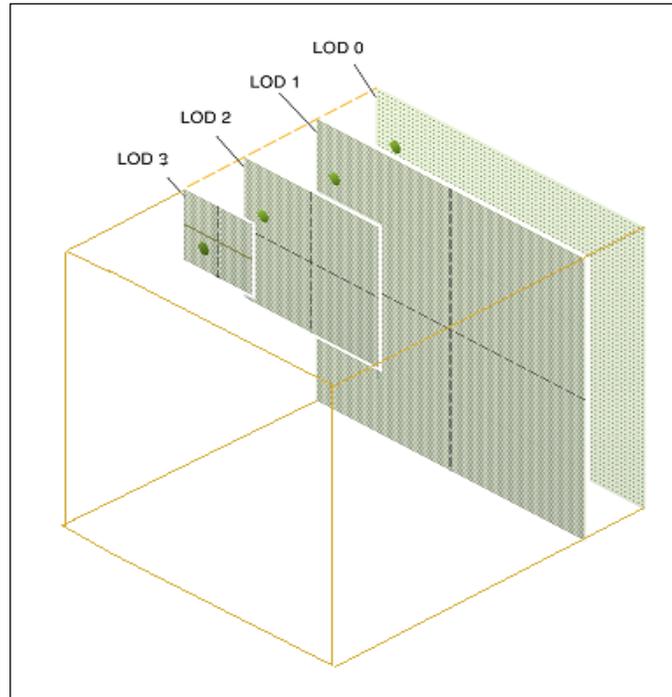


Figure 5-14: LOD Structure of Raster Datasets

The generation of each successive LOD of the MinElevation and MaxElevation components is illustrated in Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D) and again in more detail in Figure 5-16: Generation of LODs for the MinMaxElevation Dataset (2D).

The detailed algorithm for the generation of the MinMaxElevation dataset is as follows:

For a geocell, determine the finest available LOD of the Primary Terrain Elevation and Primary Alternate Terrain Elevation components, (call it $LOD = n$)

For each tile at $LOD = n$, the MinElevation (and MaxElevation) grid elements are generated by taking the corresponding minimum (and maximum) of the surrounding four “corner grid data elements” of $LOD = n$ of the Primary Terrain Elevation component (illustrated as red dots in Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D)). If the Primary Alternate Terrain Elevation component exists at $LOD = n$, the value of the Elevation needs to be taken into account because it provides a better estimate of the minimum or maximum elevation of the grid element. In other words, each MinElevation sample value represents the minimum for the area formed by the surrounding four “corner grid data elements” of the Primary Terrain Elevation plus the contribution of the Primary Alternate Terrain Elevation for the grid element. Likewise, each MaxElevation sample represents the maximum of the area formed by the surrounding four “corner grid data elements” of the Primary Terrain Elevation plus the contribution of the Primary Alternate Terrain Elevation for the grid element, illustrated as green dots in Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D).

Note that the generation of the rightmost (column) and topmost (row) of values of a tile requires access to the adjacent tiles of the Primary Terrain Elevation. Note however that the availability of Primary Elevation Data at $LOD = n$ within the entire CDB geocell cannot be guaranteed since the CDB permits the generation of the Terrain Elevation Dataset at different resolutions for each geographic area as illustrated in Figure 5-18: Availability of LODs for Elevation and MinMaxElevation Datasets.

As a result, a slight adjustment to the above algorithm is needed in order to cater to the case where Elevation data is missing in adjacent tiles. There are two cases to consider:

1. If Elevation data in the adjacent tiles (above and/or to the right) is not available at $n \geq LOD \geq -10$, then one or more of the 4 corner grid elements samples will be missing, hence will not be available to “participate” in the $\min()$ or $\max()$ function. In other words, the $\min()$ and $\max()$ functions should be designed to cater to a variable number of inputs depending on the availability of valid corner grid elements.
2. If Elevation data in adjacent tile(s) is not available at $LOD = n$ but is available at a coarser LOD (call it $LOD = m$, where $m \geq -10$), then the corner grid Elevation values of the $LOD = m$ should be propagated to finer $LOD = n$ so that they can participate in the $\min()$ or $\max()$ functions. This principle is illustrated in Figure 5-17: Generation of LODs for the MinMaxElevation Dataset (1D) – Special Case.

Each grid element value of the next coarser level-of-detail ($LOD = n-1$) of the MinElevation (and MaxElevation) dataset is generated by taking the minimum (and maximum) of four surrounding values of $LOD = n$ of the MinElevation (and MaxElevation) dataset, illustrated as red dots in Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D).

Repeat steps 2 and 3 for levels of detail $LOD = n-2, n-3$, until $LOD = -10$ is reached.

Perform step 4, but this time with $LOD = m$, ($m \geq -10$). Note that if Primary Elevation data in adjacent tile(s) is not available at $LOD = m$ but is available at a coarser LOD (call it $LOD = p$, where $p \geq -10$), then the corner grid Elevation values of the $LOD = p$ must be propagated to finer $LOD = m$ so that they can participate in the $\min()$ or $\max()$ functions.

Repeat until all LODs have been processed. Note that the MaxElevation tiles at $LOD = -10$ contain a single value which represents the highest elevation point for the entire geocell. Likewise, each of the MaxElevation tiles at $LOD = -9$ contains four values which correspond to the highest elevation points in each of the four quadrants of the corresponding geocell.

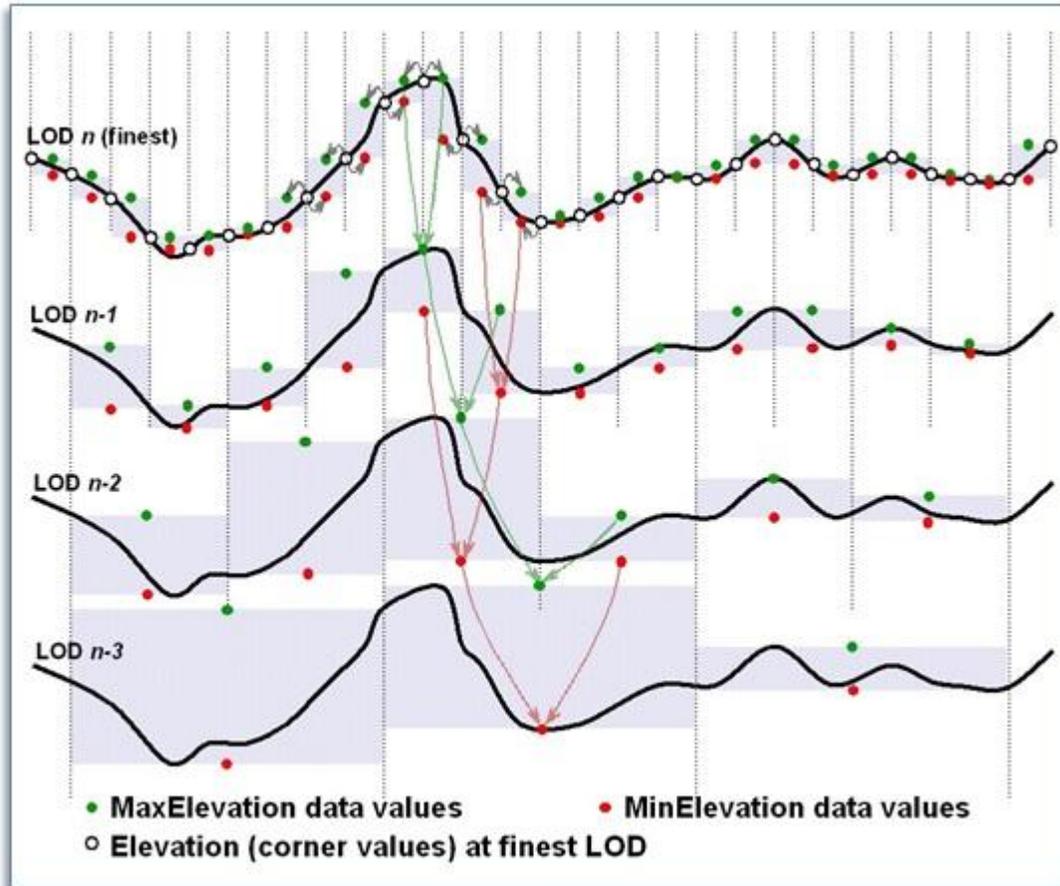


Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D)

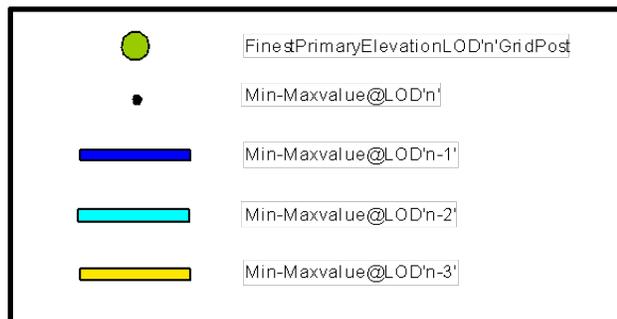
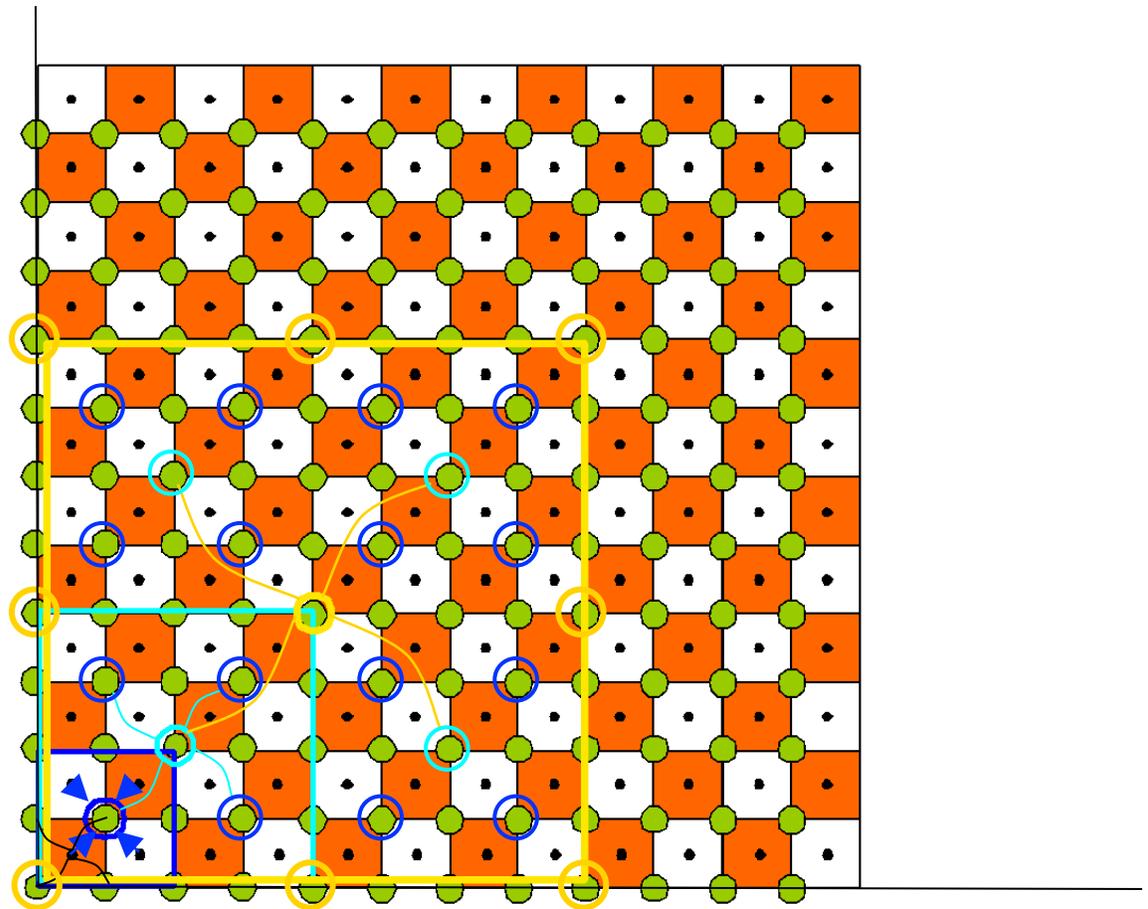


Figure 5-16: Generation of LODs for the MinMaxElevation Dataset (2D)

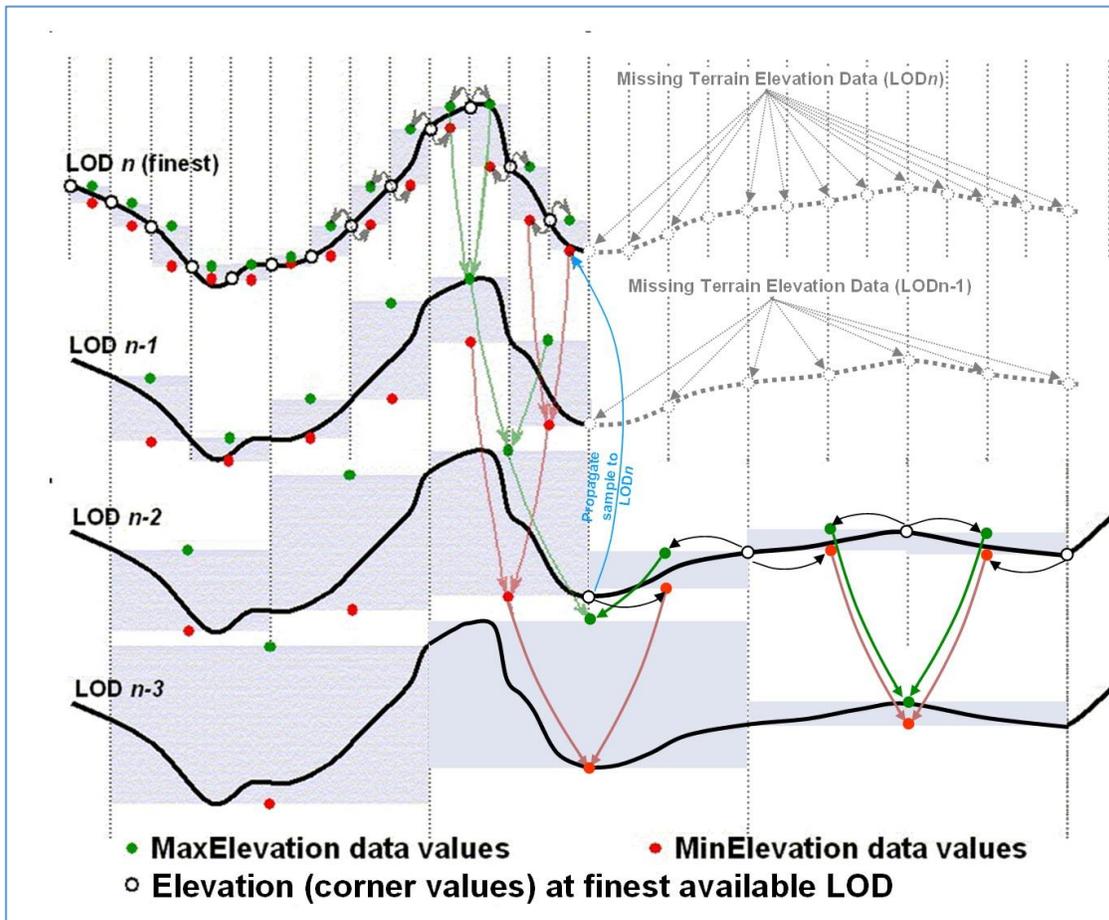


Figure 5-17: Generation of LODs for the MinMaxElevation Dataset (1D) – Special Case

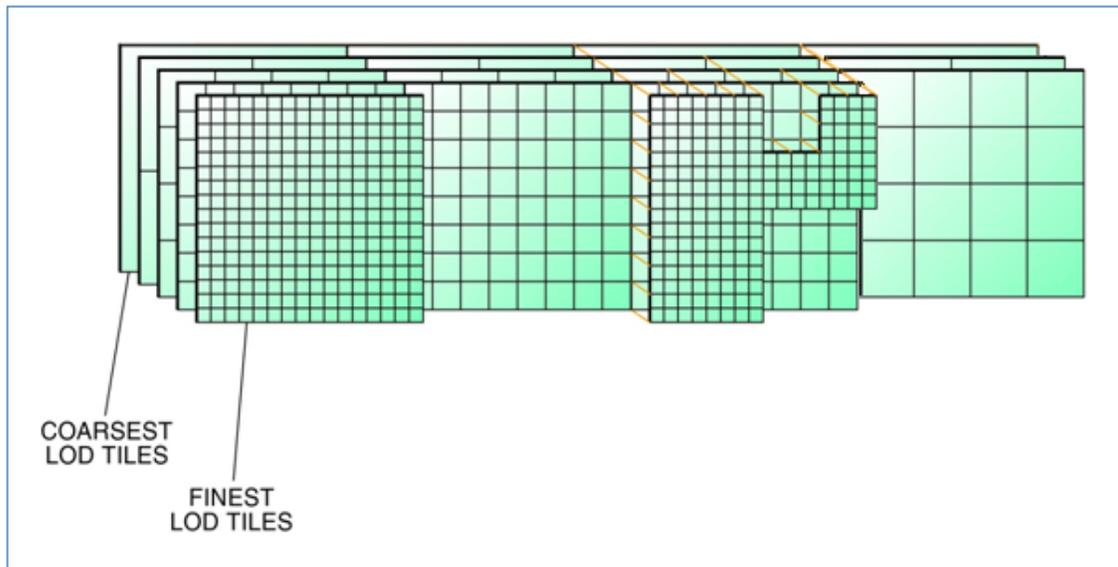


Figure 5-18: Availability of LODs for Elevation and MinMaxElevation Datasets

The CDB standard does not require that the entire LOD hierarchy be stored for the MinMaxElevation dataset. In fact, it is possible to omit some of the finest levels-of-detail from the hierarchy. The CDB Standard recommends that the MinElevation and MaxElevation need only be stored to $LOD = n - 4$ and coarser (where n is the finest available LOD of the Primary Terrain Elevation component in a geocell). For example if Primary Terrain Elevation data is available for $LOD = 15$, then the MinMaxElevation hierarchy need only be provided for $LOD = -10$ to $LOD = 11$. Note, that $LOD = -10$ to $LOD = 0$ are always required subject to the availability of Primary Terrain Elevation data (these guidelines are explained in more detail in section 5.6.1.6.4, Default Write Value).

Note that the presence of the MinMaxElevation dataset has a negligible effect on the size of the CDB. In fact, the dataset adds only 1% of additional storage over and above that required by the Primary Terrain Elevation component. This is a small price to pay in order to provide the means to significantly speed-up line-of-sight computations in applications requiring the utmost in determinism and real-time.

5.6.1.6.2 Data Type

Requirement 94

<http://www.opengis.net/spec/cdb/1.0/core/min-max-elevation-data-type>

The MinElevation and MaxElevation components *SHALL* be represented as floating-point or signed integer values. Integer values for tiles at LOD larger than 0 *SHALL* be scaled according to the following formula: $Elevation = IntValue \times 2^{-LOD}$
Integer values can make use of TIFF's 8-bit, 16-bit, or 32-bit representation.

5.6.1.6.3 *Default Read Value*

The Line-of-Sight algorithm is described in Section 8 Volume 0 - CDB Primer (Formerly Appendix A). Note that the algorithm starts with the coarsest LOD of the MinMaxElevation dataset; the algorithm recursively executes with progressively finer level-of-detail versions of the MinMaxElevation dataset until the algorithm decides it no longer needs to access finer levels or until the algorithm no longer finds finer levels of the MinMaxElevation dataset.

If none of the LODs of the MinMaxElevation dataset are provided, then simulator client-devices *SHOULD* assume default MinElevation and MaxElevation values. The default values for these datasets can be found in \CDB\Metadata\Defaults.xml and can be provided to the client-devices on demand. Handling of defaults falls under the following two cases:

- CASE I: In the case where the tile-LOD for the MinElevation and the Primary Terrain Elevation components are both missing, the CDB Specification recommends a default setting of Default_MinElevation_CaseI = Default_Elevation-1. Similarly, where a tile-LOD for MaxElevation and the Primary Terrain Elevation components are both missing, the CDB standard recommends a default setting of Default_MaxElevation_CaseI = Default_Elevation-1.
- CASE II: In the case where the tile-LOD for the MinElevation is missing and the Primary Terrain Elevation is not missing, the CDB standard recommends a default setting of Default_MinElevation_CaseII = as supplied in Defaults.xml. In the event where this default value is not supplied, the CDB standard recommends that client-devices use a default value of -400 m (corresponding to the shore of the Dead Sea) for MinElevation.

Similarly, when MaxElevation is missing and the Primary Terrain Elevation is not missing, the CDB standard recommends a default setting of Default_MaxElevation_CaseII = as supplied in Defaults.xml. In the event this default value is not supplied, the CDB standard recommends that client-devices use a default value of 8846 m (corresponding to the peak of Mount Everest) for MaxElevation.

5.6.1.6.4 *Default Write Value*

The files associated with the MinElevation and MaxElevation components for the area covered by a tile at a given LOD should not be created if the Primary Terrain Elevation data is not available.

The CDB standard recommends that the MinElevation and MaxElevation components be generated in accordance to the following guidelines:

- If the finest LOD of the Primary Terrain Elevation component is available at $\text{LOD} \geq 4$, then all LODs ranging from $-10 \leq \text{LOD} \leq (n - 4)$ of the MinElevation and MaxElevation components should be generated (where n is the finest available LOD of the Primary Terrain Elevation component). The technique illustrated in Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D) should be used to populate the LOD hierarchy. Gaps (i.e., missing levels) in the MIP-MAP hierarchy are not permitted. It is not permitted to generate MinElevation and MaxElevation components tiles that are partially populated with data.
- If the finest available LOD of the Primary Terrain Elevation component is available at $\text{LOD} \leq 3$, then all LODs ranging from $-10 \leq \text{LOD} \leq 0$ of the MinElevation and MaxElevation components should be generated. The technique illustrated in Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D) should be used to populate the LOD hierarchy. Gaps (i.e., missing levels) in the MIP-MAP hierarchy are not permitted. It is not permitted to generate MinElevation and MaxElevation components tiles that are partially populated with data.

In the event where parts of a MinElevation tile cannot be determined due to missing primary elevation tiles, the CDB standard recommends to use a default value of `Default_MinElevation_CaseIII`, -400 m (corresponding to the shore of the Dead Sea) for MinElevation. Similarly, in the event where parts of a MaxElevation tile cannot be determined due to missing primary elevation tiles, the CDB Specification recommends to use a default value of `Default_MaxElevation_CaseIII`, 8846 m (corresponding to the peak of Mount Everest) for MaxElevation. Figure 5-19: Missing MinMaxElevation Datasets shows this case:

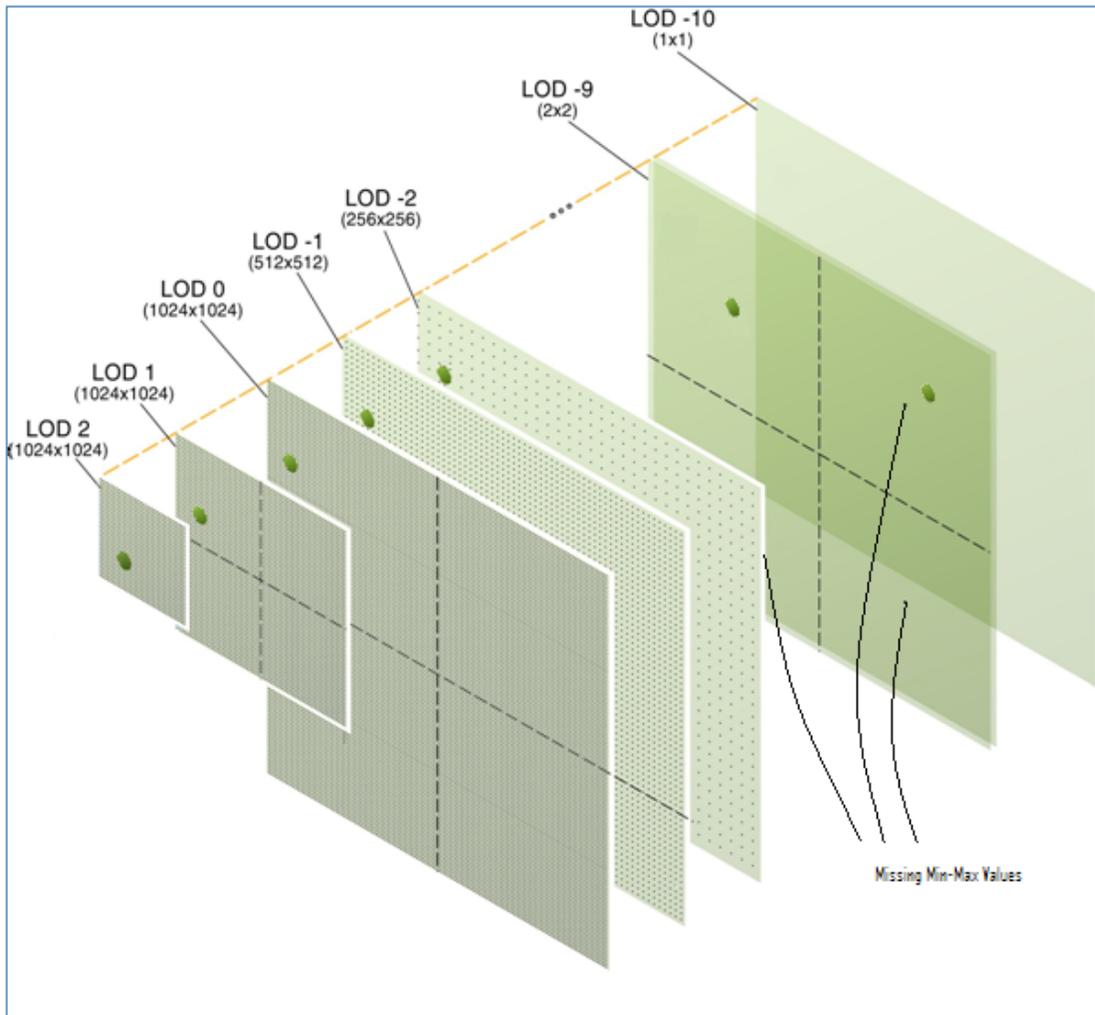


Figure 5-19: Missing MinMaxElevation Datasets

5.6.1.6.5 Supported Compression Algorithm

The CDB standard supports the LZW compression algorithm for both the MinElevation and MaxElevation components. Consider compressing the file if its content is not of type floating-point.

5.6.1.7 MaxCulture Component

The purpose of the MaxCulture component is to provide the necessary data and structure for an optimal level of determinism in the computation of line-of-sight, path finding and obstacle avoidance algorithms with the cultural features. The values of this component are based on the heights of culture features with respect to the corresponding LOD of the culture, be it its bounding sphere, its bounding box or its modeled representation (if supplied). In this context, the cultural features of the CDB are those represented by all vector tiled datasets excluding those related to NAV.

Since MinElevation, MaxElevation and MaxCulture components are defined, any line-of-sight algorithm can rapidly assess an intersection status of the line-of-sight vector with the terrain and/or with the cultural features of the CDB. Furthermore, since the MaxCulture component follows the same conventions as the MinElevation and MaxElevation components, it is easy for the LOS algorithm to combine the values to determine the highest/lowest point (with or without cultural features) in a given geographic area. The culture-variant of the LOS algorithm is virtually identical to the terrain-only case. Before undertaking its computations, the LOS algorithm must add the values of MaxCulture to the MaxElevation values, once adjusted for LOD, and then perform the first LOS determination based on this. If an intersection is detected with MaxCulture, the final determination of intersection is conducted at first with the bounding box of the cultural feature, then with the actual geometry of the cultural feature (if available).

Note that the geographic areas where MaxCulture is zero can be used to quickly identify the absence of any obstacles that can potentially affect the route of an entity.

The MaxCulture component also follows the “center grid data element” convention of the CDB Specification. In the case where a cultural feature has no modeled representation, the MaxCulture component must be generated from the feature’s bounding volume that overlaps each MaxCulture grid data element. If the feature has an associated modeled representation, the grid data of the MaxCulture component must be generated from the model geometry.

5.6.1.7.1 *Level of Details*

The coarser LODs of the MaxCulture component are iteratively derived from the finest generated LOD.

Since the MaxCulture component is intended to be used in conjunction with the MaxElevation component, it is recommended that the number of LODs for the MaxCulture component be equal or greater than the MaxElevation component.

5.6.1.7.2 *Data Type*

Requirement 95

<http://www.opengis.net/spec/cdb/1.0/core/maxculture-data-type>

The MaxCulture component *SHALL* be represented as floating-point or signed integer values. Integer values for tiles at LOD larger than 0 *SHALL* be scaled according to the following formula: $Elevation = IntValue \times 2^{-LOD}$

Integer values can make use of TIFF’s 8-bit, 16-bit, or 32-bit representation.

5.6.1.7.3 *Default Read Value*

If none of the LODs of the MaxCulture dataset are provided, then simulator client-devices should assume default MaxCulture values. The default values for these datasets can be found in \CDB\Metadata\Defaults.xml and can be provided to the client-devices on demand. Handling of defaults fall under the following two cases:

- CASE I: In the case where the MaxCulture component is missing, but there is at least one vector dataset; client-devices should assume a default MaxCulture value of `Default_MaxCulture_CaseI`. In the event this default value is not supplied, the CDB standard recommends that client-devices use a value of 600 m (corresponding to the tip of World's tallest tower plus a margin of 47 m).
- CASE II: In the case where the MaxCulture component is missing, but there is not a single vector dataset; client-devices should assume a default MaxCulture value of `Default_MaxCulture_CaseII`. In the event this default value is not supplied, the CDB standard recommends that client-devices use a value of 0 m.

5.6.1.7.4 *Default Write Value*

The files associated with the MaxCulture components for the area covered by a tile at a given LOD *should not* be created if the Primary Terrain Elevation data is not available.

The CDB standard strongly recommends that the MaxCulture dataset be generated in accordance to the following guidelines:

If the finest LOD of any vector tiled datasets is available at $LOD \geq 6$, then all LODs ranging from $-10 \leq LOD \leq (n - 6)$ of the MaxCulture dataset should be generated (where n is the finest available LOD of any vector tiled datasets). The technique illustrated in Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D) should be used to populate the LOD hierarchy. Gaps (i.e., missing levels) in the MIP-MAP hierarchy are not permitted. It is not permitted to generate MaxCulture dataset tiles that are partially populated with data.

If the finest LOD of any vector tiled datasets is available at $LOD \leq 5$, then all LODs ranging from $-10 \leq LOD \leq 0$ of the MaxCulture dataset should be generated (where n is the finest available LOD of any vector tiled datasets). The technique illustrated in Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D) should be used to populate the LOD hierarchy. Gaps (i.e., missing levels) in the MIP-MAP hierarchy are not permitted. MaxCulture dataset tiles that are partially populated with data should not be used.

5.6.1.7.5 *Supported Compression Algorithm*

The CDB standard supports the LZW compression algorithm for the MaxCulture component. Consider compressing the file if its content is not of type floating-point.

5.6.1.8 Subordinate Bathymetry Component

The Subordinate Bathymetry component consists of a grid of data values that represent the depth of water (be it of fresh water bodies or of the ocean) with respect to the corresponding data values of the Terrain Elevation (be it the Primary Terrain Elevation or Primary Alternate Terrain

Elevation components). The Subordinate Bathymetry component follows the corner grid element conventions.

The generation of Bathymetry values is best explained by the illustration found in Figure 5-20: Primary Terrain Elevation and Subordinate Bathymetry Components. In areas where Primary Terrain Elevation values correspond to the surface of a body of water, each Bathymetry value represents the height difference between the corresponding Primary Terrain Elevation value (the reference) and the Earth’s Crust. In all other areas, the Bathymetry values represent No Data³⁵. Section 6.8 of Volume 7: OGC CDB Data Model Guidance (formerly Appendix A) provides the mandated behavior of client-devices when reading a LOD of a primary component and combining it with another LOD of a subordinate component such as the Bathymetry.

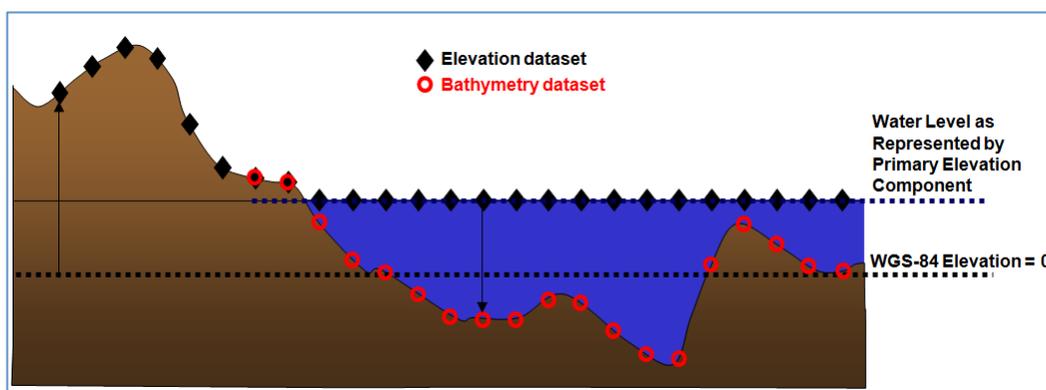


Figure 5-20: Primary Terrain Elevation and Subordinate Bathymetry Components

Positive (depth) values of Bathymetry indicate that the corresponding grid element is submerged, i.e., the Earth’s Crust is below the elevation values in the Primary Terrain Elevation component. Other values of Bathymetry (zero or negative) indicate that the corresponding grid element is not submerged.

In areas that are submerged, the Primary Terrain Elevation component represents the surface of the water, not the elevation of the Earth’s Crust. The height of any point of the Earth’s Crust with respect to the WGS-84 reference ellipsoid can be determined using Equation (eq. 5-3).

$$E_e = E - \max(0, B) \quad (\text{eq. 5-3})$$

Where E = Terrain Elevation component

³⁵ No Data is defined by a tag in the TIFF header; default is zero.

B = Bathymetry component

E_e = Earth's Crust Elevation

The resulting profile of the Earth's Crust is shown in Figure 5-21: Derived Earth Elevation Values.

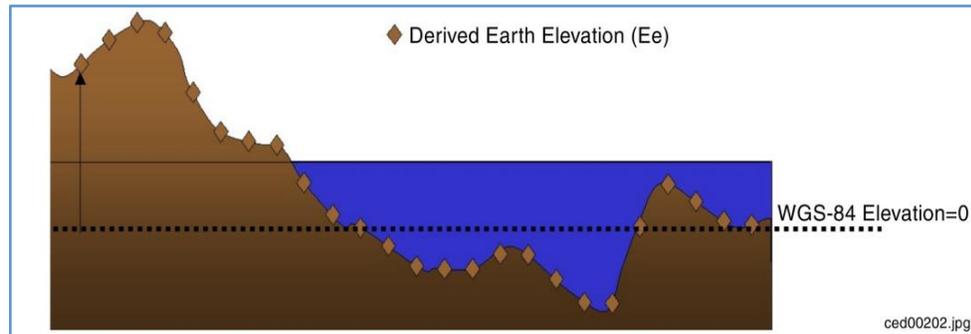


Figure 5-21: Derived Earth Elevation Values

The Bathymetry component needs to be provided only in areas on the Earth's surface where water is present. Below is another example of the relations between the Primary Elevation component and the Subordinate Bathymetry component.

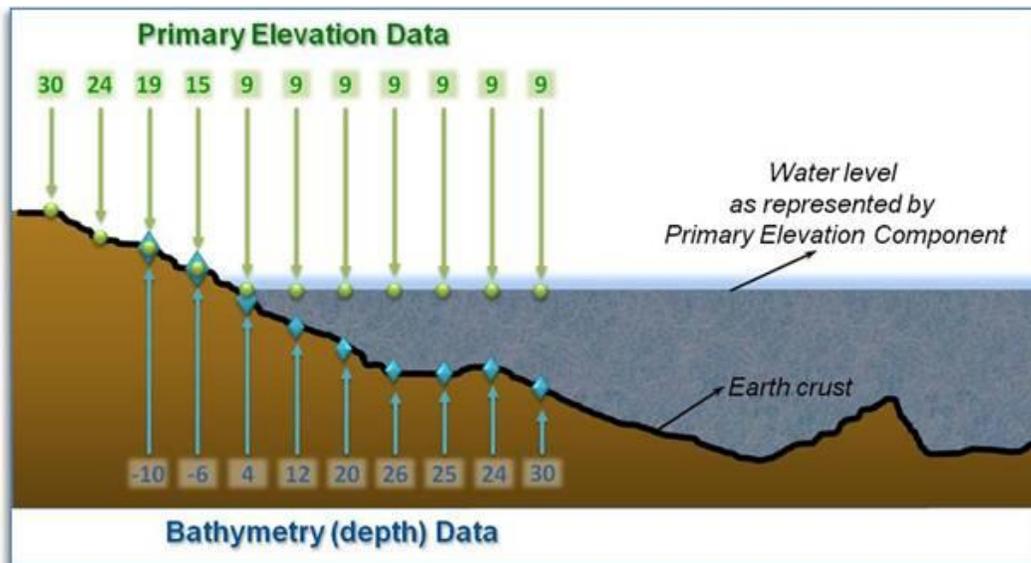


Figure 5-22: Example of Primary Terrain Elevation and Bathymetry Components

Requirements Class - Tiled Terrain Bathymetry (96-98)

</req/core/tiled-terrain-bathymetry>

Target type	Operations
Dependency	Various XML schema
Requirement 96	/req/core/bathymetry-data-type
Requirement 97	/req/core/subordinate-alternate-bathymetry-data-type
Requirement 98	/req/core/tide-component-data-type

5.6.1.8.1 Data Type

Requirement 96
http://www.opengis.net/spec/cdb/1.0/core/bathymetry-data-type
<p>The Subordinate Bathymetry component of the Elevation dataset <i>SHALL</i> be represented as a 1 or 2-channel TIFF image. The first channel contains the <i>Depth</i> of the grid post; the optional second channel indicates the <i>Type of Mesh</i> used to connect the four grid posts that are adjacent to the grid element. The depth is represented by a floating-point or signed integer value expressed in meters and relative to the Primary Terrain Elevation component. Integer values for tiles at LOD larger than 0 <i>SHALL</i> be scaled according to the following formula:</p> $\text{Depth} = \text{Intvalue} \times 2^{-\text{LOD}}$ <p>Integer values can make use of TIFF's 8-bit, 16-bit, or 32-bit representation. The <i>Mesh Type</i> <i>SHALL</i> be stored as an unsigned 8-bit integer.</p>

5.6.1.8.2 Default Read Value

Simulator client-devices should assume a *Depth* value of *Default_Bathymetry* if the data values are not available. The default value can be found in \CDB\Metadata\Defaults.xml. In the case where the default value cannot be found, the CDB standard states that client-devices use a value of zero. The default *Mesh Type* is zero.

5.6.1.8.3 Default Write Value

The files associated with the Subordinate Bathymetry component for area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with

data are not permitted. If the tool generating the Subordinate Bathymetry component does not support the optional Mesh Type, the optional second channel of the file need not be created; in which case the TIFF file becomes a single channel image.

5.6.1.8.4 *Supported Compression Algorithm*

The CDB standard supports the LZW compression algorithm for the Subordinate Bathymetry component. Consider compressing the file if its content is not of type floating-point.

5.6.1.9 Subordinate Alternate Bathymetry Component

The Subordinate Alternate Bathymetry component is similar to the Primary Alternate Terrain Elevation component: it provides a better delineation of the shoreline and bottom of water bodies such as oceans, lakes, and rivers. To do this, the Subordinate Alternate Bathymetry component encodes information that re-positions each depth samples anywhere within its assigned grid element. In other words, the “phase” of each bathymetry depth sample can be specified along the latitude and longitude axes. In effect, the Subordinate Alternate Bathymetry component provides the means to locally increase the precision of the modeled representation of the floor of water bodies. Again, it is expected that the SE tools produce the Subordinate Alternate Bathymetry component by considering constraint points, lineals and polygons provided by the modeler.

The constituents of the Subordinate Alternate Bathymetry are the depth and mesh type at the specified latitude and longitude offsets inside each grid element. These four constituents are represented as 4 channels of a TIFF image.

5.6.1.9.1 *Data Type*

Requirement 97

<http://www.opengis.net/spec/cdb/1.0/core/subordinate-alternate-bathymetry-data-type>

The first channel of the TIFF image SHALL *contain* the **Depth** component and SHALL be represented as a floating-point or signed integer value. Integer values for tiles at LOD larger than 0 are scaled according to the following formula:

$$Depth = Sample \times 2^{-LOD}$$

Integer samples can make use of TIFF’s 8-bit, 16-bit, or 32-bit representation.

The second channel of the TIFF image SHALL contain the **Mesh Type** and SHALL be stored as an unsigned 8-bit integer.

The third channel of the TIFF image SHALL contain the **Latitude Offset** and SHALL be stored as an 8-bit unsigned integer value ranging from 0 to 255. The value is scaled so

that each grid element is fragmented in 256 equal parts in the latitude direction. Thus, the grid post cannot be positioned on the latitude of the next grid element directly north of the current grid element.

The fourth channel of the TIFF image *SHALL* contain the *Longitude Offset* and *SHALL* be stored as an 8-bit unsigned integer value ranging from 0 to 255. The value is scaled so that each grid element is fragmented in 256 equal parts in the longitude direction. Thus, the grid post cannot be positioned on the longitude of the next grid element directly east of the current grid element.

5.6.1.9.2 *Default Read Value*

Simulator client-devices should assume a *Depth* of zero (as well as a *Latitude* and *Longitude Offsets* of zero and a *Mesh Type* of zero) when the Subordinate Alternate Bathymetry component is not available.

5.6.1.9.3 *Default Write Value*

The files associated with the Subordinate Alternate Bathymetry component for an area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.

5.6.1.9.4 *Supported Compression Algorithm*

The CDB standard supports the LZW compression algorithm for the Subordinate Alternate Bathymetry component. **Consider compressing the file if its content is not of type floating-point.**

5.6.1.10 *Subordinate Tide Component*

The Tide component represents the height variation of water (be it of fresh water bodies of water or of the ocean) with respect to the Primary Elevation component. The Tide component implicitly follows the corner grid element conventions. Each value in the Tide component must be matched to the available LOD elevation values of the Primary Elevation component.

The Tide component needs only to be provided in areas on the Earth's surface that are in the vicinity of water bodies. The information held in the Terrain Elevation and Tide components permits a means for client-devices to accurately determine the shoreline profile as a function of the tide level. When provided, the Tide component permits client devices to compute the elevation (with respect to the WGS-84 reference ellipsoid) in areas permanently or potentially submerged. The Tide component need not be limited to oceans; it can also be used to specify the variation of height of any body of water (rivers, lakes, gulfs, etc.).

The Tide component also permits simulation of tides that varies with location. In order to determine the shoreline profile at a given location, the simulator client-devices must first determine the height of (say) the ocean in the immediate vicinity of that location. The

sophistication of this calculation can vary greatly with simulation fidelity. A discussion of possible alternatives regarding the fidelity of simulation Tide simulation models can be found in Section 6.9 of Volume 7: CDB Data Model Guidance (formerly Appendix A).

With the CDB Tide component, simulator client-devices can readily determine the height of the ocean (or any water surface whose height varies) at any point and as a result can derive the geometry of the shoreline³⁶. While a stored vector shoreline representation might provide a more straightforward means of representing the shoreline geometry for some client-devices, that representation would not lend itself to determining the variation of the shoreline geometry with varying tides. Furthermore, a vector representation of the shoreline geometry would essentially be a single-level of detail of the shoreline geometry; as a result, it would need to be generated at a resolution designed to match the highest LOD Terrain Elevation data. Coarser shoreline LODs would essentially be samples of the shoreline vector geometry at progressively greater spatial intervals.

The CDB Tide component represents the height variation of water surfaces anywhere on the Earth's surface. The variation need not be limited to the effect of tides³⁷. The Tide component represents the height variation of the water surface above and below the mean water surface level.

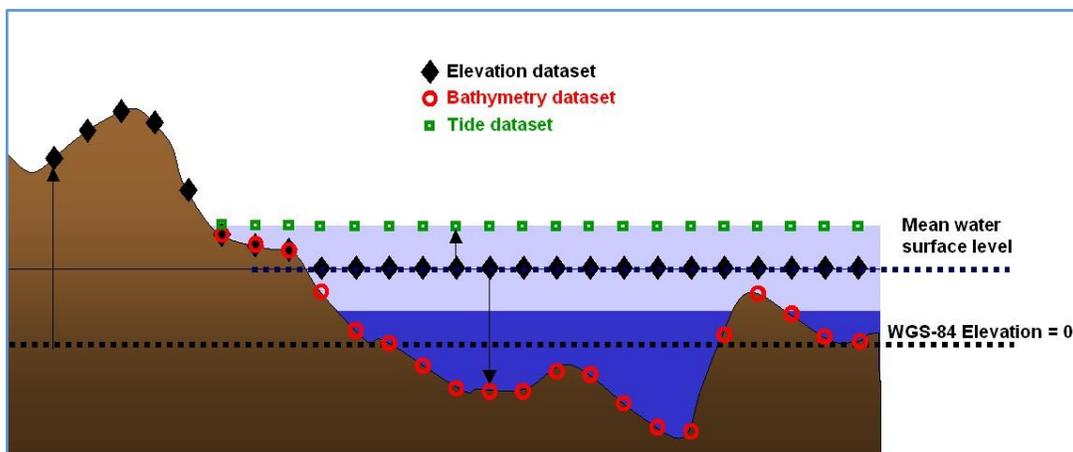


Figure 5-23: Terrain Elevation, Bathymetry and Tide Components

³⁶ While a stored vector shoreline representation might have provided a more straightforward means of representing the shoreline geometry for some client-devices, that representation was rejected because it would not lend itself to determining the variation of the shoreline geometry with varying tides.

³⁷ For instance, it could represent the nominal seasonal variations of water level of lakes and rivers.

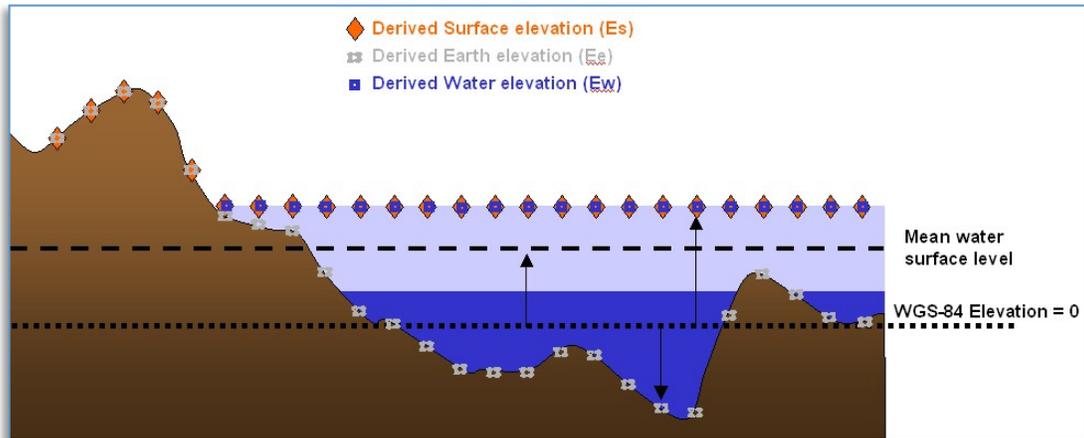


Figure 5-24: Derived Earth Elevation, Water Elevation and Surface Elevation Values

From the above components, simulation client devices can compute a) the elevation of the water E_w , b) the elevation of the earth's surface E_e (be it submerged or potentially submerged), and c) the surface elevation of the earth / water E_s . These computations can be performed in all areas where the Bathymetry and Tide components are available (e.g., areas submerged or potentially submerged). The values for E_w , E_e , and E_s are referenced to the WGS-84 mean sea-level reference level. Equation (eq. 5-4) though (eq. 5-6) can be used to compute E_w , E_e and E_s :

$$E_w = E + \min(0, B) + T \quad (\text{eq. 5-4})$$

$$E_e = E - \max(0, B) \quad (\text{eq. 5-5})$$

$$E_s = \max(E_e, E_w) \quad (\text{eq. 5-6})$$

- Where
- $E =$ Terrain Elevation component
 - $B =$ Bathymetry component
 - $T =$ Tide component
 - $E_w =$ Derived water elevation value
 - $E_e =$ Derived earth elevation value
 - $E_s =$ Derived surface elevation

Client devices interested in computing the height of the ownship over terrain or water can use equation (eq. 5-7).

$$HAT = O - E_s \quad (\text{eq. 5-7})$$

- Where
- $O =$ Ownship Altitude

Finally, client devices interested in determining the depth of water D , can use equation (eq. 5-8).

$$D = \min(0, E_w - E_e) \quad (\text{eq. 5-8})$$

NOTE: A computed value of D of 0 means the point is above water.

5.6.1.10.1 Data Type

Requirement 98
http://www.opengis.net/spec/cdb/1.0/core/tide-component-data-type
Tide components <i>SHALL</i> be represented as floating-point or signed integer values. Integer values for tiles at LOD larger than 0 <i>SHALL</i> be scaled according to the following formula: $\text{Tide} = \text{Intvalue} \times 2^{-\text{LOD}}$
Integer values can make use of TIFF's 8-bit, 16-bit, or 32-bit representation.

5.6.1.10.2 Default Read Value

Simulator client-devices should assume default Tide values if the data values are not available (files associated with the Tide component for the area covered by a tile, at a given LOD or coarser, are either missing or cannot be accessed). The default value can be provided to the client-devices on demand. The CDB standard recommends a default tide value of 2.5m (published average magnitude of tides worldwide).

Simulator client-devices should assume a default Tide value of Default_Tide if the data values are not available (files associated with the Tide component for the area covered by a tile, at a given LOD or coarser, are either missing or cannot be accessed). The default value can be found in \CDB\Metadata\Defaults.xml and can be provided to the client-devices on demand. In the case where the default value cannot be found, the CDB standard recommends that client-devices use a default tide value of 2.5m (average magnitude of tides worldwide).

5.6.1.10.3 Default Write Value

The files associated with the Tide component for area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.

5.6.1.10.4 Supported Compression Algorithm

The CDB standard supports the LZW compression algorithm for the Tide component. Consider compressing the file if its content is not of type floating-point.

5.6.2 Tiled Imagery Dataset

In a CDB compliant data store, the terrain imagery is depicted on a grid at regular geographic intervals. Each of the components of the Imagery Dataset corresponds to the raster imagery draped over the terrain skin derived from the Primary Terrain Elevation Dataset. The Raster Imagery Dataset implicitly follows the center grid element conventions.

The CDB standard defines a set of alternate terrain imagery representations corresponding to the visible spectrum terrain imagery at different periods of the year. Together, these representations are stored in a set of Visible Spectrum Terrain Imagery (VSTI) components. Each of these representations can be either monochrome or color.

In addition, the CDB standard defines a subordinate light map representation that can be applied to the selected VSTI component for a night-time representation of lighting patterns created by the projection of light-sources onto the terrain surface. The light-map can be either monochrome or color.

5.6.2.1 Raster-Based Imagery File Storage Extension Naming

As briefly mentioned earlier in Section 1.4.10, the CDB standard introduces the notion of support for JPEG 2000 raster-based storage format for raster imagery files. Since the CDB standard enforces a unique filename for each dataset file, a different file extension is required for such a dataset file format to distinguish it from TIFF for other raster based datasets, thus any raster imagery dataset shall be stored under the “.jp2” file extension.

5.6.2.1.1 JPEG 2000 Metadata

In addition to the compressed image data, the JPEG 2000 files may contain metadata to hold additional data boxes. They are the Intellectual Property box, XML box, URL box and UUID box. Among them, the XML box is perfectly suited to store formatted metadata concerning the source of this data, or the security attributes associated with the file usage. Below is the XML format description of such metadata to be supported as part of a CDB implementation. It is to be noted that the existence of this XML metadata box does not contain any information necessary for decoding the image portion, and the correct interpretation of the XML data will not change the visual appearance of the image. This metadata is divided in two distinct elements, namely ORIGIN and SECURITY.

Requirements Class - Tiled JPEG metadata(99-100)
--

/req/core/tiled-raster-jpeg-metadata
--

Target type	Operations
Dependency	Various XML schema
Requirement 99	/req/core/jpeg-origin-metadata
Requirement 100	/req/core/jpeg-security-metadata

5.6.2.1.1.1 Origin of data

Requirement 99			
http://www.opengis.net/spec/cdb/1.0/core/jpeg-origin-metadata			
The rules for metadata about the origin of data defined in the following table <i>SHALL</i> be implemented with a JPEG image in the CDB data store.			
XML Tag Name	Format	Description	Values
datetime	STRING	File Date & Time: This field <i>SHALL</i> contain the file's origination in the format CCYYMMDDhhmmss, where CC is the first two digits of the year (00-99), YY is the last two digits of the year (00-99), MM is the month (01-012), DD is the day (01-31), hh is the hour (00-23), mm is the minute (00-59), and ss is the second (00-59). UTC is assumed to be the time zone designator to express the time of day.	Default is 14 spaces Date Format: CCYYMMDDhhmmss
originatingstationid	STRING	Originating Station ID: This field <i>SHALL</i> contain the identification code or name of the originating organization, system, station, or product. It shall not be filled with spaces.	This 10-character field <i>SHALL NOT</i> be blank
originatorname	STRING	Originator's Name: This field <i>SHALL</i> contain a valid name for the operator who originated the file. If the field is all spaces, it shall represent that no operator is assigned responsibility for origination.	Default is 24 spaces
originatorphone	STRING	Originator's Phone Number. This field <i>SHALL</i> contain valid phone number for the operator who originated the file. If the field is all spaces, it shall represent that no phone number is available for the operator assigned responsibility for origination.	Default is 18 spaces
originatororganizati	STRING	Originator's Organization. This field	Default is 80 spaces

on		<i>SHALL</i> contain a valid name of the supporting organization.	
originatoraddress	STRING	Originator's Address. This field <i>SHALL</i> contain a valid address of the supporting organization.	Default is 256 spaces
originatoremail	STRING	Originator's Electronic Mail Address. This field <i>SHALL</i> contain a valid email address of the supporting organization.	Default is 100 spaces
originatorwebsite	STRING	Originator's Web Site Address. This field shall contain a valid web site address of the supporting organization.	Default is 100 spaces
originatorremark	STRING	Originator's Remark Text. This field <i>SHALL</i> contain description text for any special remarks concerning the file.	Default is 100 spaces

5.6.2.1.1.2 Security

Requirement 100			
http://www.opengis.net/spec/cdb/core/jpeg-security-metadata			
The rules for metadata about the security of data defined in the following table <i>SHALL</i> be implemented with a JPEG image in the CDB data store. The rules for using JPEG security are defined in the table below.			
Attribute Name	Format	Description	Values
classificationlevel	BYTE	File Security Classification: This field <i>SHALL</i> contain a 1-character valid value representing the classification level of the entire file.	Valid values are: T (=Top Secret), S (=Secret), C (=Confidential), R (=Restricted), U (=Unclassified).
system	STRING	File Security Classification System: This field <i>SHALL</i> contain valid values indicating the national or multinational security system used to classify the file. If this field is all blank spaces, it shall imply that no security classification system applies to the file.	Default is 2 spaces Country Codes per FIPS 10-4 shall be used to indicate national security systems; codes found in DIAM 65-19 shall be used to indicate multinational security systems.
codewords	STRING	File Codewords: This field <i>SHALL</i> contain a valid indicator of the security compartments associated with	Default is 11 spaces

		the file.	
controlhandling	STRING	File Control and Handling. This field <i>SHALL</i> contain valid additional security control and/or handling instructions (caveats) associated with the file. Values include digraphs found in DIAM 65-19 and/or MIL_STD_2500B-Table A-4. The digraph may indicate single or multiple caveats. The selection of a relevant caveat(s) is application specific. If this field is all spaces, it shall imply that no additional control and handling instructions apply to the file.	Default is 2 spaces Values include one or more of the tri/digraphs found in DIAM 65-19 and/or MIL_STD_2500B-Table A-4. Multiple entries shall be separated by a single space
releaseinstructions	STRING	File Releasing Instructions. This field <i>SHALL</i> contain a valid list of country and/or multilateral entity codes to which countries and/or multilateral entities the file is authorized for release. If this field is all spaces, it shall imply that no file release instructions apply.	Default is 20 spaces Valid items in the list are one or more country codes as found in FIPS 10-4 and/or codes identifying multilateral entities as found in DIAM 65-19.
declassificationtype	STRING	File Declassification Type. This field <i>SHALL</i> contain a valid indicator of the type of security declassification or downgrading instructions which apply to the file. If this field is all spaces, it shall imply that no file security declassification or downgrading instructions apply.	Default is 2 spaces Valid values are: DD (=declassify on a specific date), DE (=declassify upon occurrence of an event), GD (=downgrade to a specified level on a specific date), GE (=downgrade to a specified level upon occurrence of an event), O (=OADR), X (= exempt from automatic declassification).
declassificationdate	STRING	File Declassification Date. This field <i>SHALL</i> indicate the date on which a file is to be declassified if the value in File Declassification Type is DD. If this field is all spaces, it shall imply that no file	Default is 8 spaces Date Format: CCYYMMDD

		declassification date applies.	
declassificationexemption	STRING	<p>File Declassification Exemption. This field <i>SHALL</i> indicate the reason the file is exempt from automatic declassification if the value in File Declassification Type is X.</p> <p>If this field is all spaces, it shall imply that a file declassification exemption does not apply.</p>	<p>Default is 4 spaces</p> <p>Valid values are X1 through X8 and X251 through X259. X1 through X8 correspond to the declassification exemptions found in DOD 5200.1-R, paragraphs 4-202b(1) through (8) for material exempt from the 10-year rule. X251 through X259 correspond to the declassification exemptions found in DOD 5200.1-R, paragraphs 4-301a(1) through (9) for permanently valuable material exempt from the 25-year declassification system.</p>
filedowngrade	BYTE	<p>File Downgrade. This field <i>SHALL</i> indicate the classification level to which a file is to be downgraded if the values in File Declassification Type are GD or GE. If this field is all spaces, it <i>SHALL</i> imply that file security downgrading does not apply.</p>	<p>Default is 1 space</p> <p>Valid values are: S (=Secret), C (=Confidential), R (=Restricted).</p>
filedowngradedate	STRING	<p>File Downgrade Date. This field <i>SHALL</i> indicate the date on which a file is to be downgraded if the value in File Declassification Type is GD. If this field is all spaces, it <i>SHALL</i> imply that a file security downgrading date does not apply.</p>	<p>Default is 8 spaces</p> <p>Date Format: CCYYMMDD</p>
classificationtext	STRING	<p>File Classification Text. This field <i>SHALL</i> be used to provide additional information about file classification to include identification of declassification or downgrading event if the values in File Declassification Type are DE or GE. It may also be used to identify multiple classification sources and/or any other special handling rules. If this field is all spaces, it <i>SHALL</i> imply that</p>	<p>Default is 43 spaces</p> <p>Values are user defined free text.</p>

		additional information about file classification does not apply.	
classificationauthoritytype	BYTE	File Classification Authority Type. This field <i>SHALL</i> indicate the type of authority used to classify the file. If this field is all spaces, it <i>SHALL</i> imply that file classification authority type does not apply.	Default is 1 single space Valid values are: O (= original classification authority), D (= derivative from a single source), M (=derivative from multiple sources).
classificationauthority	STRING	File Classification Authority: This field <i>SHALL</i> identify the classification authority for the file dependent upon the value in File Classification Authority Type. If this field is all spaces, it <i>SHALL</i> imply that no file classification authority applies.	Default is 40 spaces Values are user defined free text which should contain the following information: - original classification authority name and position or personal identifier if the value in File Classification Authority Type is O; - title of the document or security classification guide used to classify the file if the value in File Classification Authority Type is D; and Derive-Multiple if the file classification was derived from multiple sources. In the latter case, the file originator will maintain a record of the sources used in accordance with existing security directives. One of the multiple sources may also be identified in File Classification Text if desired.
classificationreason	BYTE	File Classification Reason: This field <i>SHALL</i> contain values indicating the reason for classifying the file. If this field is all spaces, it <i>SHALL</i> imply that no file classification reason applies.	Default is 1 single space Valid values are A through G. These correspond to the reasons for original classification per E.O. 12958, Section 1.5.(a) through (g).
classificationsourcedate	STRING	File Security Source Date: This field <i>SHALL</i> indicate the date of the source used to derive the classification of the file. In the case of multiple	Default is 8 spaces Date Format: CCYYMMDD

		sources, the date of the most recent source <i>SHALL</i> be used. If this field is all spaces, it <i>SHALL</i> imply that a file security source date does not apply.	
controlnumber	STRING	File Security Control Number: This field <i>SHALL</i> contain a valid security control number associated with the file. The format of the security control number <i>SHALL</i> be in accordance with the regulations governing the appropriate security channel(s). If this field is all spaces, it <i>SHALL</i> imply that no file security control number applies.	Default is 15 spaces
filecopynumber	INT	File Copy Number: This field <i>SHALL</i> contain the copy number of the file. If this field is all zeros, it <i>SHALL</i> imply that there is no tracking of file's number of copies.	Default is 00000 Number can range between: 00000 to 99999
numberofcopies	INT	File Number of Copies: This field <i>SHALL</i> contain the total number of copies of the file. If this field is all zeros, it <i>SHALL</i> imply that there is no tracking of numbered file copies.	Default is 00000 Number can range between: 00000 to 99999

5.6.2.1.1.3 JPEG 2000 XML Example

Table 5-14: XML Tags for the JPEG 2000 Metadata

```
<?xml version="1.0" encoding="UTF-8"?>
<JP2METADATA name="JPEG2000XML"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="JP2MetaData.xsd">
  <ORIGIN>
    <datetime>
    </datetime>
    <originatingstationid>
    </originatingstationid>
    <originatorname>
    </originatorname>
    <originatorphone>
    </originatorphone>
    <originatororganization>
    </originatororganization>
    <originatoraddress>
    </originatoraddress>
    <originatoremail>
    </originatoremail>
    <originatorwebsite>
    </originatorwebsite>
    <originatorremark>
    </originatorremark>
  </ORIGIN>
  <SECURITY>
    <classificationlevel>
    </classificationlevel>
    <system>
    </system>
    <codewords>
    </codewords>
    <controlhandling>
    </controlhandling>
    <releaseinstructions>
    </releaseinstructions>
    <declassificationtype>
    </declassificationtype>
    <declassificationdate>
    </declassificationdate>
    <declassificationexemption>
    </declassificationexemption>
    <filedowngrade>
    </filedowngrade>
    <filedowngradedate>
    </filedowngradedate>
    <classificationtext>
    </classificationtext>
    <classificationauthoritytype>
    </classificationauthoritytype>
    <classificationauthority>
    </classificationauthority>
    <classificationreason>
    </classificationreason>
    <securitysourcedate>
    </securitysourcedate>
    <controlnumber>
    </controlnumber>
    <filecopynumber>
    </filecopynumber>
    <numberofcopies>
    </numberofcopies>
  </SECURITY>
</JP2METADATA>
```

5.6.2.2 List of all Imagery Dataset Components

The Imagery Dataset is comprised of several components listed here and detailed in the subsequent sections.

Table 5-15: Imagery Dataset Components

CS1	CS2	File Extension	Component Name	Component Description
Dataset 004, Imagery				
001	001	*.jp2	Yearly VSTI Representation	Corresponds to the terrain imagery draped (orthographically) over the terrain skin derived from the Primary Terrain Elevation Dataset. This is the preferred Dataset Component for year-round representative terrain imagery. It may be single-channel monochrome or 3-channel color image. This Dataset Component

				follows the center grid conventions. Can be used interchangeably with all other Alternate VSTI representations.
002	001..004	*.jp2	Seasonal VSTI Representations	Deprecated – Replaced with Quarterly VSTI Representations below
003	001..012	*.jp2	Monthly VSTI Representations	Monthly equivalent of Yearly VSTI representation, i.e., this is the preferred Dataset Component for month-based representative terrain imagery. Can be used interchangeably with all other Alternate VSTI representations.
004	001..004	*.jp2	Quarterly VSTI Representations	Equivalent to Yearly VSTI representation but for the selected quarter of the year. Can be used interchangeably with all other Alternate VSTI representations.
005	001	*.jp2	Subordinate VSTLM	Corresponds to the terrain light maps draped (orthographically) over the terrain skin derived from the Primary Terrain Elevation Dataset. It may be single-channel monochrome or 3-channel color image. This Dataset Component follows the center grid conventions.

5.6.2.3 Visible Spectrum Terrain Imagery (VSTI) Components

The VSTI component provides the visible spectrum imagery that is geographically draped (and usually ortho-rectified) over the geometric representation of the terrain skin that is stored in the Primary Terrain Elevation Dataset. The CDB standard provides the means to (optionally) store alternate representations of the terrain imagery in order to provide the simulation client-devices terrain representations that best represent the time-of-year being simulated. There are three alternate approaches to the generation and storage of the VSTI Imagery Dataset and they are organized as follows:

- **Yearly:** The first approach requires a single, year-round representation of the terrain imagery.
- **Quarterly:** The second approach requires four variants of the terrain imagery, one per calendar-year quarter³⁸.
- **Monthly:** The third approach requires monthly-variants of the terrain imagery, one per month.

The VSTI Imagery Datasets can be provided and stored in any combination, be it yearly, quarterly and/or monthly.

The VSTI dataset implicitly follows the center grid element conventions.

³⁸ Each quarter corresponds to specific months of the year. This concept of calendar-year quarters is distinct from the concept of seasons whereby the later depends on whether the user is in the northern or the southern hemisphere of Earth.

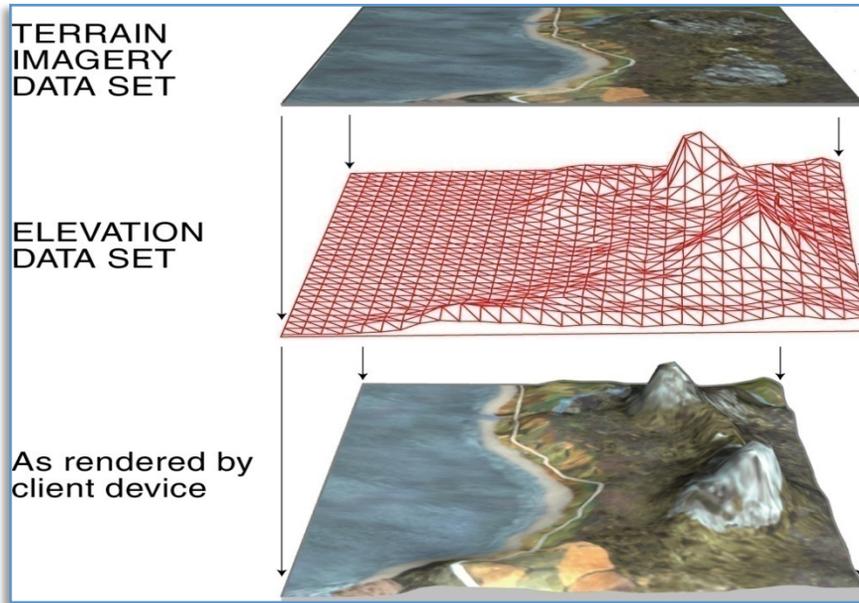


Figure 5-25: Projection of Terrain Imagery Dataset onto Terrain Elevation Dataset

The CDB grid representation of this raster imagery assumes a gamma of 1.0 (see Volume 0: OGC CDB Companion Primer for the CDB standard: Model and Physical Database Structure, Section 9) and a color space model in conformance with Windows sRGB or YUV Color Space Profile. sRGB is the default color space in Windows, based on the IEC 61966-2-1 Standard. CDB terrain imagery can optionally be compressed into JPEG 2000 with varying degrees of quantization (quality) levels. However, if using a quantization level different than 0, lossy image results in possible image degradation and artifact addition.

Requirements Class - VSTI (101-102)	
/req/core/tiled-raster-vsti	
Target type	Operations
Dependency	Various XML schema
Requirement 101	/req/core/vsti-component-data-type
Requirement 102	/req/core/client-vsti-texture-selection-rules

5.6.2.3.1 Data Type

Requirement 101

<http://www.opengis.net/spec/cdb/1.0/core/vsti-component-data-type>

The VSTI component *SHALL* be represented as single-channel gray-scale images, or as triple-channel non-paletted color images in JPEG 2000 format. The use of transparency on terrain imagery is not allowed.

5.6.2.3.2 Default Read Value

Simulator client-devices *should* default the VSTI values if the data values are not available (files associated with the VSTI dataset for the area covered by a tile, at a given LOD or coarser, are either missing or cannot be accessed). The default value can be found in \CDB\Metadata\Defaults.xml and can be provided to the client-devices on demand. In the case where the default value cannot be found, the CDB standard recommends that client-devices use a default value of half-intensity (0.5). Note that the default values are expressed as floating-point numbers ranging from 0.0 to 1.0. This ensures that the default is interpreted in a consistent manner independently of the data representation in the *.jp2 file.

Requirement 102

<http://www.opengis.net/spec/cdb/1.0/core/client-vsti-texture-selection-rules>

Simulation client-devices *SHALL* select the VSTI texture that best represents the simulation date. The retrieval of VSTI textures by the client-devices *SHALL* follow the following conventions:

The simulation date *SHALL* be converted to a month of the year

If the monthly VSTI representation for that month number is absent, then the client-device *SHALL* determine which quarter of the year it is and search for the quarterly representation of the VSTI

If a quarterly representation is absent, then the client-device *SHALL* search for a yearly representation of the VSTI

If the yearly representation is absent, then the client-device *SHALL* default to the Yearly default values found in \CDB\Metadata\Defaults.xml as follows:

- a. Default_VSTI_Y_Mono
- b. Default_VSTI_Y_Red
- c. Default_VSTI_Y_Green
- d. Default_VSTI_Y_Blue

The above conventions are summarized in Table 5-16.

Table 5-16: VSTI Default Read Values

Monthly		Quarterly	Yearly	Default
January	001	001	001	Default_VSTI_Y_Mono Default_VSTI_Y_Red Default_VSTI_Y_Green Default_VSTI_Y_Blue
February	002			
March	003			
April	004	002		
May	005			
June	006			
July	007	003		
August	008			

September	009			
October	010	004		
November	011			
December	012			

5.6.2.3.3 *Default Gamma Correction*

The default gamma correction is defined by `Default_Imagery_Gamma` found in the `Defaults.xml` metadata file. If `Default_Imagery_Gamma` is not found in `Defaults.xml`, or if `Defaults.xml` is not found in the metadata directory, assume a default gamma correction of 1.0.

5.6.2.3.4 *Default Write Value*

The files associated with the VSTI component for area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.

5.6.2.3.5 *Supported Compression Algorithm*

The CDB standard supports a compressed form using the JPEG 2000 format.

5.6.2.4 Visible Spectrum Terrain Light Map (VSTLM) Component

The VSTLM component provides the visible spectrum terrain light maps that are orthographically draped over the terrain skin (e.g., Primary Terrain Elevation Dataset) and onto T2DModels. In addition, client-devices can also use the VSTLM component to orthographically project the light map onto GTModels, GSModels and statically-positioned MModels.

Light maps fall under the category of subordinate textures. The light maps are used in low illumination conditions (dusk, dawn, night) to represent the combined illumination effect of man-made light sources (primarily lamp-posts) on the terrain. The technique provides a convenient

means to produce interesting and entirely predictable lighting effects without resorting to computationally intensive local light sources.

The light map adds to the lighting levels provided by the simulated ambient light level; the combined ambient lighting and the light map together modulate the underlying VSTI. Light maps can be created in a number of ways, either manually with a tool such as Photoshop, from night-time imagery or finally from an off-line rendering process that simulates the illumination effect of the urban lighting sources onto the terrain.

Requirements Class - VSTLM (103)	
/req/core/tiled-raster-vstlm	
Target type	Operations
Dependency	Various XML schema
Requirement 103	/req/core/vstlm-component-data-type

5.6.2.4.1 Data Type

Requirement 103
http://www.opengis.net/spec/cdb/1.0/core/vstlm-component-data-type
The VSTLM component <i>SHALL</i> be represented as single-channel gray-scale images, or as triple-channel color images. The data <i>SHALL</i> be stored in JPEG 2000 format. Note that in the case of a monochrome VSTLM, the implied chrominance of the VSTLM is white.

5.6.2.4.2 Default Read Value

Simulator client-devices *should* default the VSTLM values if the data values are not available (files associated with the VSTLM dataset for the area covered by a tile, at a given LOD or coarser, are either missing or cannot be accessed). The default value can be found in \CDB\Metadata\Defaults.xml and can be provided to the client-devices on demand. In the case where the default value cannot be found, the CDB standard recommends that client-devices use a default value of zero-intensity (0.0). Note that the default values are expressed as floating-point

numbers ranging from 0.0 to 1.0. This ensures that the default is interpreted in a consistent manner independently of its representation in the *.jp2 file. The default values are:

- Default_VSTLM_Mono
- Default_VSTLM_Red
- Default_VSTLM_Green
- Default_VSTLM_Blue

5.6.2.4.3 *Default Gamma Correction*

The default gamma correction is defined by Default_Imagery_Gamma found in the Defaults.xml metadata file. If Default_Imagery_Gamma is not found in Defaults.xml, or if Defaults.xml is not found in the metadata directory, assume a default gamma correction of 1.0.

5.6.2.4.4 *Default Write Value*

The files associated with the VSTLM component for area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.

5.6.2.4.5 *Supported Compression Algorithm*

The CDB standard supports compressed form using the JPEG 2000 format.

5.6.3 Tiled Raster Material Dataset

Historically, Digital Feature Analysis (DFAD) and VPF (Vector Product Format) data have been used to provide the terrain and cultural content information used by the real-time sensors, the computer generated forces and the visual systems. The vectorized outlines of areas tagged with attribution data had a cartoon-like appearance because they did not capture the richly varying mixture of materials. Each geometric shape would be represented as a single material type resulting in simplistic sensor scenes. Sometimes, a locally applied texture pattern would be applied to add some realism to the single material type. While it is still possible to build a CDB compliant data store in this manner, the preferred approach involves the use of the Raster Material Dataset described here. The Raster Material Dataset can be readily derived from the (image) classification of mono, color or multispectral imagery. This Dataset is a material-coded image. It is independent of wavelength (visible, infrared, etc.) and is designed to support geospecific, multi-spectral scene simulation across any computing platform. The Raster Material Dataset is typically generated from material classification and mixing analysis (see Figure 5-26: Image Classification Example). It can be developed directly from geospecific imagery, (e.g., SPOT, Landsat) and have a one-to-one correspondence with the image data. The Raster Material Dataset results in a smoothly varying simulation data store free of hard edges characteristically found in vectorized DFAD outlines.

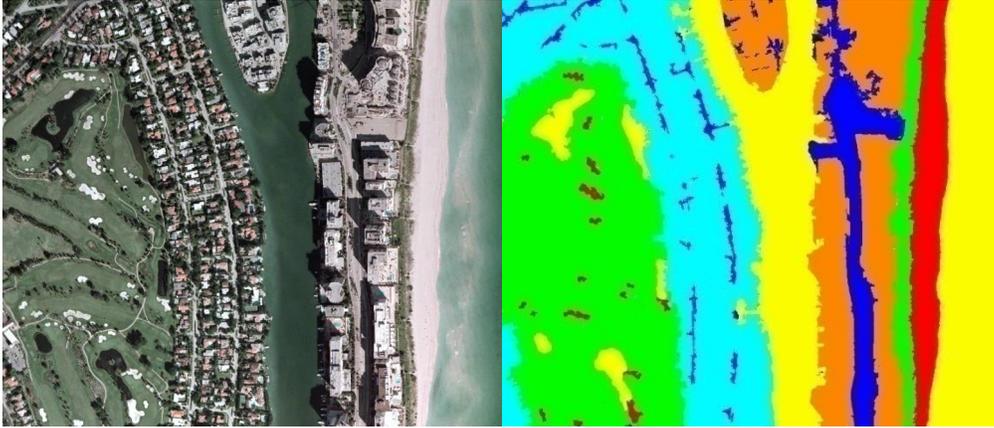


Figure 5-26: Image Classification Example

The Raster Material Dataset provides the means to store the types of materials and the area coverage of each material within each pixel of the dataset. In all other aspects, it follows conventions similar to the VSTI dataset.

A Raster Material Dataset consists of a set or stack of “*n*” Material Layers. This stacking arrangement permits the modeler to assign up to “*n*” materials to the area covered by each pixel of the Raster Material Dataset. The Raster Material Dataset also consists of a stack of “*n-1*” mixture layers; the mixture layers define the proportions of materials at each pixel. The CDB standard makes provision for up to 255 materials, (i.e., any pixel within the Raster Material Dataset can be assigned up to 255 materials).

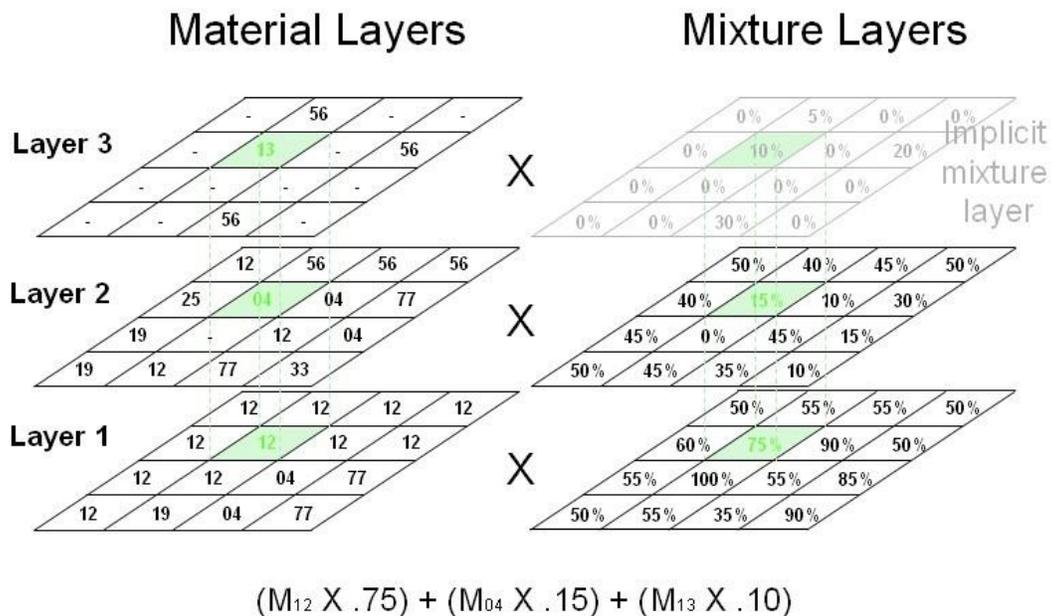


Figure 5-27: Example of a Raster Material Dataset

Figure 5-23: Terrain Elevation, Bathymetry and Tide Components, provides an example of a Raster Material Dataset consisting of 3 material layers and two associated mixture layers. The first Material Layer (i.e., layer 1) consists of a regular grid of pixels; each pixel contains a code that represents the (composite) material with largest area coverage. Likewise, the second Material Layer consists of a grid of pixels whose code represents the (composite) material with second-highest area coverage. Additional layers are added until the area corresponding to the combined area of all (composite) materials at each pixel sums to 100%. Note that in layer 2, the material layer value of some pixels can be ignored (shown as “-“ in the illustration) because layer 1 had a material mixture value of 100%. Similarly, the material layer value of some pixels in layer 3 can also be ignored because the mixture layers 1 and 2 add to 100%. In these cases, the CDB Standard recommends that the layer value be assigned a Default_Material_Layer value of 0.

NOTE: The numeric value for Default_Material_Layer is zero (“0”) and is reserved by the CDB standard.

Mixture Layers represent the percentage area coverage of each material within each pixel of each mixture. Since all layers must add to 100%, it is possible to represent “*n*” Material Layers with a set of “*n-1*” Mixture Layers. The last layer is implicit, and it is set to (100% - Sum of areas from previous layers). In the case where there is a single Material Layer, there is no need to store the (implicit) Mixture Layer. When there are two or more Material Layers, the Mixture Layer(s) should be generated.

5.6.3.1 List of all Raster Material Dataset Components

The Raster Material Dataset is comprised of several components listed here and detailed in the subsequent sections.

Table 5-17: Raster Material Dataset Components

CS1	CS2	File Extension	Component Name	Component Description
Dataset 005, RMTexture				
001	001..255	*.tif	Composite Material Index	Each texel is an index into the Composite Material Table (dataset 006). CS2 is the layer number. Corresponds to a 2D grid of composite material indices draped (orthographically) over the terrain skin derived from the Primary Terrain Elevation Dataset.
002	001..254	*.tif	Composite Material Mixture	Each texel indicates the proportion (between 0.0 and 1.0) of the composite material found in the corresponding material layer. CS2 is the layer number. Corresponds to a 2D grid draped (orthographically) over the terrain skin derived from the Primary Terrain Elevation Dataset. This Dataset component follows the center grid conventions.
Dataset 006, RMDescriptor				
001	001	*.xml	Composite Material Table	The Composite Material Table is referenced by the Composite Material Index component and contains the definitions of the composite materials of a Tile-LOD.

5.6.3.2 Composite Material Index Component

As mentioned earlier, the CDB standard allows pixels of the Raster Material Dataset to consist of several (up to 255) composite materials. To accomplish this, it uses a layering concept that permits the assignment of several composite materials for each pixel in the Material Dataset. As a result, the chosen representation for the Raster Material Dataset consists of a set or stacks of “*n*” Material Layers, where “*n*” is the maximum number of composite materials encountered in any pixel of the CDB tile at the specified LOD.

The code assigned to each pixel of each Material Layer is the index of a Composite Material found in the Terrain Composite Material Table (TCMT) defined in 5.6.3.4.

Each pixel of the first Material Layer (e.g., layer “1”) consists of a code that represents the composite material with largest area coverage for that pixel. Likewise, each pixel of the second Material Layer consists of a code that represents the composite material with second-highest area coverage for that pixel. Additional layers are added until the area corresponding to the combined area of all composite materials at each pixel sums to 100%.

Requirements Class - Raster Composite Material (104-106)

[/req/core/tiled-raster-composite-material](#)

Target type	Operations
Dependency	Various XML schema
Requirement 104	/req/core/material-layer-data-type
Requirement 105	/req/core/material-mixture-data-type
Requirement 106	/req/core/tcmt-data-type

5.6.3.2.1 Data Type

Requirement 104
http://www.opengis.net/spec/cdb/1.0/core/material-layer-data-type
The Material Layer components each <i>SHALL</i> be represented as single-channel, material coded one byte unsigned integer value images stored in TIFF.

5.6.3.2.2 Default Read Value

If none of the Material Layer components are available (files associated with the Material Layer dataset for the area covered by a tile, at a given LOD or coarser, are either missing or cannot be accessed), simulator client-devices should default to a single Material Layer component whose content defaults to a single default Composite Material. The default Composite Material can be found in \CDB\Metadata\Defaults.xml and can be provided to the client-devices on demand. The default value is:

- Default_Material_Layer (0)

In the case where the default value cannot be found, the CDB Specification recommends that client-devices default to single substrate composite material whose base material is:

- Default_Base_Material (BM_LAND-MOOR)

5.6.3.2.3 Default Write Value

The files associated with the Material Layer components for the area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.

5.6.3.2.4 Supported Compression Algorithm

The LZW compression algorithm is applicable to Composite Material Index component in the TIFF file format.

5.6.3.3 Composite Material Mixture Component

A Mixture Layer accompanies each Material Layer; its dimensions are identical to those of the Material Layer. The pixel values of the Mixture Layer “*n*” represent the area coverage of Material Layer “*n*”. Since all layers must add to 100%, it is possible to represent “*n*” Material Layers with a set of “*n-1*” Mixture Layers. As a result, the last layer is implicit, and it is set to (100% - Sum of areas from previous layers). In the case where there is a single Material Layer, there is no need to store the (implicit) Mixture Layer. When there are two or more Material Layers, the Mixture Layer(s) must be generated.

5.6.3.3.1 Data Type

Requirement 105

<http://www.opengis.net/spec/cdb/1.0/core/material-mixture-data-type>

The Material Mixture components each *SHALL* be stored in a single-channel TIFF file. All values range from 0.0 (0%) to 1.0 (100%). Integral types represent scaled integers to fit the range 0.0 to 1.0; floating-point values are limited to the range 0.0 to 1.0.

5.6.3.3.2 Default Read Value

If none of the Material Mixture components are available (files associated with the Material Mixture dataset for the area covered by a tile, at a given LOD or coarser, are either missing or cannot be accessed), simulator client-devices should assume equal mixing for all available Material Layers.

5.6.3.3.3 Default Write Value

The files associated with the Material Mixture components for the area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.

5.6.3.3.4 Supported Compression Algorithm

The LZW compression algorithm is applicable to Material Mixture components in the TIFF file format. Consider compressing the file if its content is not of type floating-point.

5.6.3.4 Composite Material Table Component

This Composite Material Table is called the Terrain CMT, or just TCMT; it provides a list of the Composite Materials shared by the Material Layers of the Material Dataset. There is one TCMT for each CDB tile.

5.6.3.4.1 Data Type

Requirement 106 Composite Materials Table (CMT)

<http://www.opengis.net/spec/cdb/1.0/core/tcmt-data-type>

The TCMT *SHALL* follow the syntax described in Section 2.5.2.2, Composite Material Tables (CMT).

5.6.3.4.2 *Default Read Value*

Simulator client-devices should default the Terrain Composite Material Table if file associated with the Terrain Composite Material Table for the area covered by a tile, at a given LOD, is either missing or cannot be accessed. The default values for the Terrain Composite Material Table can be found in \CDB\Metadata\Defaults.xml and can be provided to the client-devices on demand. The default value is a single Composite Material and is named: Default_Material_Layer (0)

If the default information cannot be found within the \CDB\Metadata\Defaults.xml file, the CDB standard recommends defaulting to single substrate composite material whose base material is named: Default_Base_Material (BM_LAND-MOOR)

If an index is not found in the Terrain Composite Material Table, use the same defaulting mechanism.

5.6.3.4.3 *Default Write Value*

The files associated with the Terrain Composite Material Table for the area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.

5.7 Tiled Vector Datasets

Vector tiles differ from their raster counterpart in three important ways. First of all, a vector tiles internal structure permits a non-uniform distribution of elements within the tile, (i.e., the position of each element within the tile is explicit). Secondly, the vector tiles internal structure permits a variable number of elements within the tile confines. Finally, it is possible to control the distribution of the element types from a single list.

Conceptually, the LOD of a vector tile implicitly provides the average density of elements within the tile. The run-time level-of-detail behavior that controls the rendered number of data elements depends on various parameters and on the off-line filtering process.

NOTE: The LOD referred to in this section concerns itself with the grouping of cultural features into tiles at specified LODs, and not with the geometric accuracy or detail of the modeled representation of these features.

Requirements Class - Tiled Vector Datasets (107-112)	
/req/core/tiled-vector-datasets	
Target type	Operations
Dependency	Various XML schema
Requirement 107	/req/core/vector-type-rule
Requirement 108	/req/core/vector-client-origin-with-model
Requirement 109	/req/core/model-light-point-position
Requirement 110	/req/core/light-point-with-z-position
Requirement 111	/req/core/vertex-position

5.7.1 Introduction to Vector Datasets

A point feature is a geographic entity where its simplest representation resolves to a point with general attributes such as size, position, or material. A lineal feature is a geographic entity that defines a one-dimensional feature such as a road, a canal, a river. A polygon feature is a geographic entity where its simplest representation resolves to a two-dimension feature such as a lake or soil type boundary. In this context, a geographic entity is always specified by latitude and longitude coordinates; in turn, the geographic entity is conformed onto the terrain by the client-device.

The lineal and polygon feature's representation abstractly resolves to a one or a two dimensional feature. Unless otherwise specifically mentioned, lineal and polygon feature's representations are not used to model a geometrical representation. However, these features may optionally reference an explicitly modeled representation (for example an OpenFlight model) located in the geospecific model or the geotypical model datasets.

The current CDB standard specifies Esri Shapefiles to represent vector data and attributes. All shape types are supported to represent point, line, and polygon features. Please refer to Volume 4: OGC CDB Best Practice use of Shapefiles for Vector Data Storage for detailed information and requirements for using Shapefiles for storing vector data in a CDB data store.

Allowed vector data features types for a CDB data store are as follows:

Value	Vector type	Fields
0	Null shape	None
1	Point	X, Y
3	Polyline	MBR, Number of parts, Number of points, Parts, Points

5	Polygon	MBR, Number of parts, Number of points, Parts, Points
8	MultiPoint	MBR, Number of points, Points
11	PointZ	X, Y, Z <i>Optional: M</i>
13	PolylineZ	<i>Mandatory:</i> MBR, Number of parts, Number of points, Parts, Points, Z range, Z array <i>Optional: M range, M array</i>
15	PolygonZ	<i>Mandatory:</i> MBR, Number of parts, Number of points, Parts, Points, Z range, Z array <i>Optional: M range, M array</i>
18	MultiPointZ	<i>Mandatory:</i> MBR, Number of points, Points, Z range, Z array <i>Optional: M range, M array</i>
21	PointM	X, Y, M
23	PolylineM	<i>Mandatory:</i> MBR, Number of parts, Number of points, Parts, Points <i>Optional: M range, M array</i>
25	PolygonM	<i>Mandatory:</i> MBR, Number of parts, Number of points, Parts, Points <i>Optional: M range, M array</i>
28	MultiPointM	<i>Mandatory:</i> MBR, Number of points, Points <i>Optional Fields: M range, M array</i>
31	MultiPatch	<i>Mandatory:</i> MBR, Number of parts, Number of points, Parts, Part types, Points, Z range, Z array <i>Optional: M range, M array</i>

Requirement 107 Vector Type

<http://www.opengis.net/spec/cdb/1.0/vector-type-rule>

All instances of the feature *SHALL* be of the same vector data type. The CDB standard requires a maximum of one vector data type for point features, a maximum of one type for lineal features and a maximum of one type for polygon features for each tile (for a maximum of 3 feature vector files per tile).

All of the information that is needed to instance features is organized in accordance to the CDB tile structure. All the tiled Vector dataset files are located in the same directory; the dataset's second component selector (CS2) is used to differentiate between files with the same extension or with the same Vector features. Table 5-18: Component Selector 2 for Vector Dataset, presents the list of codes that are allocated. Note that Vector datasets do not necessarily use all of the Dataset Component Selector 2 reserved codes. Users of the CDB standard should refer to the appropriate section for an enumeration of the supported File Component Selector 2 codes as well as the ones specific to the Dataset.

The Vector dataset concept and the feature concepts overlap somewhat; some of the Vector datasets are generalizations or specializations of feature codes. Section 1.5 provides a recommended mapping of the feature attributes across the CDB compliant datasets. Note that the same feature *should* not have two representations.

Table 5-18: Component Selector 2 for Vector Datasets

CS2	Dataset Component Name	Supported Vector Types
001	Point features	Point, PointZ, PointM, MultiPoint, MultiPointZ, MultiPointM
002	Point feature class-level attributes	N/A
003	Lineal features	PolyLine, PolyLineZ, PolyLineM
004	Lineal feature class-level attributes	N/A
005	Polygon features	Polygon, PolygonZ, PolygonM, Multipatch
006	Polygon feature class-level attributes	N/A
007	Lineal figure point features	Point, PointZ, PointM, MultiPoint, MultiPointZ, MultiPointM
008	Lineal figure point feature class-level attributes	N/A
009	Polygon figure point features	Point, PointZ, PointM, MultiPoint, MultiPointZ, MultiPointM
010	Polygon figure point feature class-level attributes	N/A
011	2D relationship tile connections	N/A
012	Deprecated	N/A
013	Deprecated	N/A
014	Deprecated	N/A

015	2D relationship dataset connections	N/A
016	Point feature extended-level attributes	N/A
017	Lineal feature extended-level attributes	N/A
018	Polygon feature extended-level attributes	N/A
019	Lineal Figure Point extended-level attributes	N/A
020	Polygon Figure Point extended-level attributes	N/A

5.7.1.1 Vector Type Usage and Conventions

This section establishes conventions globally applicable to the usage of all vector features.

For explicitly modeled point cultural features:

Each point-feature of a CDB database can be optionally associated with a GSModel, a GTModel or MModel. The rendering of GSModels, GTModels or MModels by client-devices requires an associated point-feature. The linkage is made through point-feature attributes which together provide the information needed by client-devices to locate the Model from the appropriate Dataset at the appropriate level-of-detail. The following feature attributes provide the necessary linkage:

- **FeatureCode-FSC**: Feature Code and Subcode
- **MODL**: Model Name
- **MODT**: Model Type
- **MLOD**: Model Level-of-Detail
- **MMDC**: Moving Model DIS Code

Requirement 108 Client device model origin and orientation

<http://www.opengis.net/spec/cdb/1.0/core/vector-client-origin-with-model>

If the feature has an associated model, client-devices *SHALL* position the model's origin at the specified coordinate, orient the model's Y-axis in accordance to the AO1 attribute, and align the model's Z-axis so that it points up. In the case of vector data types that do not have a Z component value, the object's height value *SHALL* be referenced to the underlying terrain; as a result, client-devices are required to position the model's origin wrt underlying terrain elevation dataset. For Point features with a Z component, client-devices *SHALL* position the model as per the AHGT class attribute value. If AHGT is true, the model's origin *SHALL* be positioned to the value specified by the Z component (Absolute Terrain Altitude), irrelevant of the terrain elevation dataset. If AHGT is false or not present, the model's origin *SHALL* be positioned to the value specified by the underlying terrain offset by the Z component value.

For modeled light points:

It is common practice within the simulation industry to model light points without their associated support structures. In this case, the preferred way to model light points is through the use of point-features within the Airport and Environmental Light-Point Features Datasets of a CDB data store; consequently, there are no Models associated with Airport and Environmental Light-Point Features.

Note however that is entirely permissible to also model lights points with their associated support structures. In this case, the OGC CDB Rules for Encoding Data using OpenFlight (Volume 6) representing the support structure also contains light points as specified in section 6.11, Model Light Points.

The “modeling” of light points is accomplished via the following light-point feature attributes:

- ***LTYP***: Light Type
- ***LPH***: Light Phase
- ***AOI***: Angle of Orientation

Requirement 109 Light Points CRS
http://www.opengis.net/spec/cdb/1.0/core/model-light-point-position
The position of light points <i>SHALL</i> be expressed using WGS-84 geographic coordinates (latitude, longitude, altitude), as explained in Volume 8 OGC CDB Spatial Reference System Guidance. Client-devices <i>SHALL</i> position the center of the light point at the specified coordinate, orient directional light points in accordance to the AO1 attribute.

The elevation angle component of a directional light point is intrinsic to its type (for instance a VASITypeT\2.5_Degree\Fly-Up1_light should be used to represent a Type VASI light used for a 2.5 degree glide slope). In the case of vector data types that do not have a Z component value, the light point’s height value is referenced to the underlying terrain; as a result, client-devices are required to elevate the light point’s center with regard to the underlying terrain elevation dataset.

Requirement 110 Client device position for Light Point with Z
http://www.opengis.net/spec/cdb/1.0/core/light-point-with-z-position
For Light Point features with a Z component, client-devices <i>SHALL</i> position the light point’s center as per the AHGT value. If AHGT is true, the light point’s center <i>SHALL</i> be positioned to the value specified by the Z component (Absolute Terrain Altitude),

irrelevant of the terrain elevation dataset. If AHGT is false or not present, the light point's center *SHALL* be positioned to the value specified by the underlying terrain offset by the Z component value.

For point, lineal and polygon features that are not modeled:

The CDB data model does not make mandatory that all features of the CDB be modeled; as a result, each feature is *optionally* associated with a GSModel, a GTModel or a MModel.

Requirement 111 Vertex CRS

<http://www.opengis.net/spec/cdb/1.0/core/vertex-position>

The position of vertices *SHALL* be expressed using WGS-84 geographic coordinates (latitude, longitude, altitude), as explained in Volume 8 OGC CDB Spatial Reference System Guidance. In the case of vector data types that do not have a Z component value, the vertex's height value *SHALL* be referenced to the underlying terrain; as a result, client-devices are required to position the vertex's origin wrt underlying terrain elevation dataset. For vector data types with a Z component, client-devices *SHALL* position the vertex as per the AHGT value. If AHGT is true, the vertex *SHALL* be positioned to the value specified by the Z component (Absolute Terrain Altitude), irrelevant of the terrain elevation dataset. If AHGT is false or not present, the vertex *SHALL* be positioned to the value specified by the underlying terrain offset by the Z component value.

The AHGT attribute, when present, is always ignored when the Z component value does not exist.

The bounding box coordinates Xmin, Ymin, Xmax, Ymax required by some vector data types are expressed using WGS-84 geographic coordinates (in accordance with Volume 8: OGC CDB Spatial Reference System Guidance).

The value of M and Mrange found in some of the vector types (PointM, MultiPointM, PolygonM, and PolyLineM) is ignored by client-devices.

5.7.1.2 CDB Attribution

Attributes are used to describe one or more real or virtual characteristics of a feature. Features can be assigned a variable number of attributes.

Requirements Class - Vector datasets Mandatory Attribute Usage (113-116)	
/req/core/tiled-vector-datasets-mandatory-attributes	
Target type	Operations
Dependency	Various XML schema
Requirement 113	/req/core/mandatory-attribute-compliance
Requirement 114	/req/core/optional-attribute-compliance
Requirement 115	/req/core/dataset-instance-schema
Requirement 116	/req/core/attributes-metadata

5.7.1.2.1 *Attribute Code*

A unique four-digit numeric code is associated to each attribute. For example, the attribute "Angle of Orientation" has an attribute code of "0003".

5.7.1.2.2 *Attribute Identifier*

A unique three-character or four-character alphanumeric identifier is associated to the attributes that are governed by this standard. Attributes other than those governed by the CDB standard may not have an assigned identifier. For example, the CDB attribute "Length" has the "LEN" identifier. The identifier is a case-sensitive string of up to 10 characters. In the case of instance-level and class-level attributes, the identifier is used as the base name for a column name, such as counties.dbf, in a vector attribute encoding.

5.7.1.2.3 *Attribute Semantics*

Each attribute is associated with a textual description (describing semantic information), which provides a human readable definition of the attribute.

5.7.1.2.4 *Attributes Values*

A value can be assigned to each attribute. The data type, length, format, range, usage, units, compatibility and schema of each attribute value is governed by this standard. Attribute values give quantitative/qualitative meaning to the attribute.

5.7.1.2.5 *Attribute Usage*

CDB attribution usage falls in the following categories:

Requirement 113 Mandatory Attribute Usage

<http://www.opengis.net/spec/cdb/1.0/core/mandatory-attribute-compliance>

A CDB compliant vector dataset *SHALL* have no missing mandatory attributes. See Section 5.7.1.3 for Mandatory attributes and their characteristics.

Clarification note Mandatory: A mandatory attribute is an attribute whose value *SHALL* be provided for all of the features of a specified dataset, i.e., a producer of CDB data (e.g., tools) is required to generate values for mandatory attributes. Consumers of CDB compliant data (tools and/or simulator client-devices) can rely on the availability of mandatory attributes.

Recommended: A recommended attribute is an attribute whose value should be provided for all of the features of a specified dataset. Consumers of CDB compliant data (tools and/or simulator client-devices) can always rely on the availability of recommended attributes since the attribute value is either provided explicitly in the CDB data store/dataset or provided implicitly as a defaulted value in accordance to section 5.7.1.3, CDB Attributes. A CDB with defaulted recommended attributes is considered compliant by this standard; however, the performance of one or more of the client-devices (commonly found on simulation devices) may be adversely affected.

Optional: An optional attribute is an attribute whose value may (optionally) be provided for all of the features of a specified dataset. Consumers of CDB compliant data (tools and/or simulator client-devices) cannot rely on the availability of optional attributes.

Requirement 114 Missing optional attributes

<http://www.opengis.net/spec/cdb/1.0/core/optional-attribute-compliance>

A CDB dataset with missing optional attributes *SHALL* be considered compliant by this standard; however, the performance of one or more of the client-devices (commonly found on simulation devices) may be enhanced by including the optional attributes. See Section 5.7.1.3 for Optional attributes and their characteristics.

Dependent: A dependent attribute is an attribute whose value depends on another attribute, be it mandatory, recommended, or optional. The attribute is considered mandatory if the attribute it depends on is mandatory. Likewise, the attribute is considered recommended if the attribute it depends on is recommended. Finally, the attribute is considered optional if the attribute it depends on is optional.

Note that attribute usage information for each of the CDB attributes can be found in section 5.7.1.3, CDB Attributes and in Table 5-27: Allocation of CDB Attributes to Vector Datasets

5.7.1.2.6 Attribution Data Compatibility

The CDB standard provides a flexible means to tag features with attribution data. The CDB standard accommodates the vast majority of attribution data that is in use today and available through formats and products supported by the NGA and other US governmental agencies. The CDB standard provides the means to attribute features with attribution data with varied origins.

5.7.1.2.6.1 CDB Attributes

CDB attributes are attributes whose semantics, data type, length, format, range, usage, units, compatibility and schema are entirely governed by the CDB standard. Most of these attributes are unique to the CDB standard, i.e., these attributes are not found in source data that conforms to various (US) governmental standards and specifications. As a result, this attribution data must be derived via CDB tool automation or provided directly by the user.

5.7.1.2.6.2 Geomatics Attributes

Geomatics attributes are attributes whose semantics, data type, length, format, range, usage, and units, are governed by various governmental/industrial specifications and standards. Such attributes are generally found in source data that conforms to such standards and specifications. While the CDB standard itself does not define and govern the usage of these attributes, it nonetheless accommodates their storage within the repository structure of a CDB compliant dataset/data store.

5.7.1.2.6.3 Vendor Attributes

Vendor attributes are attributes whose semantics, data type, length, format, range, usage, and units are governed by one or more vendors. In general, such attributes cannot be used by other vendors since they are often proprietary. Such attributes exclude the above two types of attributes and are generally unique to each vendor. While the CDB standard itself does not define and govern the usage of these attributes, it nonetheless accommodates their storage within a CDB compliant data store/dataset.

5.7.1.2.7 Attribution Schemas

The CDB standard offers three different attribution schemas. Each of the schemas offers different trade-offs in the manner attribution data is accessed and stored. Each of these schemas is largely motivated by the storage size considerations, and flexibility in the manner attribution data can be assigned to individual features and to groups of features.

The three attribution schemas supported by the CDB standard are:

- Instance-level schema
- Class-level schema
- Extended-level schema

5.7.1.2.7.1 Instance-level Schema

This is the attribution schema used for features whose attributes and attribute values vary with each instance of a feature in a dataset.

Requirement 115 Dataset instance schema

<http://www.opengis.net/spec/cdb/1.0/core/dataset-instance-schema>

The attributes and their values *SHALL* be specified as attribution columns in the instance-level attribute file (or table) that accompanies the dataset's vector/geometry file. This attribute file *SHALL* be referred to as the Dataset Instance-level attribute file (or table).

Each instance of a feature is characterized by a corresponding set of instance-level attributes implemented as a row within the instance-level attribute file (or table). Each attribute *SHALL* be uniquely defined by an attribute identifier that is a "case-sensitive" character string of 10 characters or less. This 10-character limitation of attribute names is for backwards compatibility with the dBASE III+ File format structure implementation (see Volume 2: OGC CDB Core Model and Physical Structure: Annexes, Annex E).

The data type, length, format, range, usage, and units of the attribution values are specific to each attribute. The interpretation of the attribution data value is governed by metadata that describes the data type, the data format, the allowable range of the data, the numerical precision of the data, the units associated with the data, etc for each attribute.

Requirement 116 Attributes Metadata

<http://www.opengis.net/spec/cdb/1.0/core/attributes-metadata>

The CDB_Attributes.xml metadata file *SHALL* be used to describe all the CDB attributes listed in 5.7.1.2, CDB Attribution. This attribute metadata *.xml file *SHALL* be included in the CDB folder hierarchy under the CDB Metadata directory (refer to section 3.1.1, Metadata Directory). The CDB_Attributes.xml metadata file *SHALL* be based on a *.xsd schema file that governs the syntax and structure of the attribute metadata file.

Refer to Section 1.4.2 of this document for a listing of the attribute metadata schema.

Each row of this instance-level file (or table) contains the instance-level attribute values for a corresponding feature in the *.shp file. The first column of each row within the instance-level *.dbf is always the classname (CNAM). If the classname is not used, its value is set to blank, and all of the classname attributes need to be added to the instance-level file (or table). The

number of columns in a Dataset Instance-level file (or table) is different for each dataset. All of the instance-level attributes are CDB attributes.

Dataset *.shp File	Dataset Instance-level *.dbf								
	CNAM	A01	LENL	LPN	RTAI	SCALX	SCALY	SCALZ	⋮
record 1	-	5.2°	1000 m	1	89%	1.0	1.0	1.0	...
record 2	-	15.0°	2500 m	1	71%	1.5	1.5	1.5	...
record 3	-	90.0°	565 m	1	99%	1.0	1.0	1.0	...
record 4	-	82.0°	1003 m	1	85%	1.0	1.0	1.0	...
.	-
.	-
.	-

Figure 5-28: Instance-level Attribution Schema

5.7.1.2.7.2 Class-level Schema

This is the preferred attribution schema for features whose attributes and attribute values can be shared by one or more of the instances of a feature in a dataset.

The attributes and their values are logically re-grouped under a classname (CNAM attribute) that stands for the group of attributes specific to that class. Each row of the class-level *.dbf file corresponds to a classname found in the instance-level *.dbf shape file. Each attribute class is characterized by a set of attributes implemented as a row within the class-level *.dbf file. In turn, each attribute is uniquely defined by a name that is a “case-sensitive” character string of 10 characters or less. This 10-character limitation of attribute names is set for backwards compatibility due to use of the dBASE III+ File format structure (see Volume 2: OGC CDB Core Model and Physical Structure: Annexes, Annex E).

The interpretation of the attribution data value is governed by metadata that describes the data type, the data format, the allowable range of the data, the numerical precision of the data, the units associated with the data, etc for each attribute. The CDB_Attributes.xml metadata file is used to describe all the CDB attributes listed in section 5.7.1.2, CDB Attribution. The CDB_Attributes.xml file must be included in the CDB folder hierarchy under the CDB Metadata directory (refer to section 3.1.1, Metadata Directory). The CDB_Attributes.xml metadata file is structured in accordance to a *.xsd schema file. Refer to Section 1.4.2 of this document for a description of the attribute metadata schema.

The first column of the file is the classname and acts as the primary key to access table entries; all other rows correspond to the attributes represented by the classname. All of the class-level attributes are CDB attributes. For each dataset, a classname is unique within a geocell.

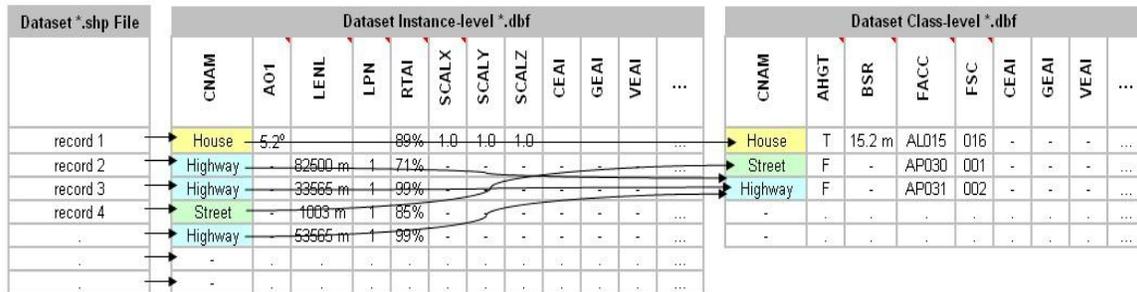


Figure 5-29: Class-level Attribution Schema

5.7.1.2.7.3 Extended-level Schema

The CDB standard provides an alternate attribution schema that can be used (in many cases) to supplement the instance-level and class-level schemas.

The extended-level schema can be used to represent CDB attributes, Geomatics attributes and Vendor attributes. However, the extended-level schema is the only means by which Geomatics attributes and Vendor attributes can be accessed.

Linkage to the extended-level CDB attribution data is accomplished through the CEAI attribute; CEAI is an index to a link list of CDB attributes stored in the extended-level attribute file (or table)³⁹. Similarly, the GEAI and VEAI attributes are also indices to link lists of attributes stored in the extended-level attribute file (or table). The extended-level files has the structure described in section 5.7.1.2.7.4, Structure of Extended-level Files.

There is one attribute metadata file (named CDB_Attributes.xml) that describes the CDB attributes of section 5.7.1.3, CDB Attributes, one attribute metadata file (named Geomatics_Attributes.xml) for the Geomatics attributes and one attribute metadata file (named Vendor_Attributes.xml) for the Vendor attributes. All three attribute metadata *.xml files are optional; if provided, they are included in the CDB folder hierarchy under the CDB Metadata directory (refer to section 3.1.1, Metadata Directory). All three attribute metadata *.xml files share the same schema. The schema that governs the contents of the attribute metadata files is Vector_Attributes.xsd. Refer to Section 1.4.2 of this document for a description of this schema.

5.7.1.2.7.4 Structure of Extended-level Attribute Files or Tables

Each row of the Extended-Level attribute files (or tables) correspond to an attribute. Each attribute row consists of four columns as follows:

Column 1 – **LNK (Link)**: The first column is a numeric 6-digit index to the next entry of a link list of attributes (a value of 0 marks the end of the list). The LNK field provides a means to organize attributes into link lists of attributes that in turn can be associated with a feature.

Column 2 – **GRP (Group)**: The second column provides an 8-character string that is used to name the group to which the extended attributes belongs to. The actual value of this character

³⁹ Currently a *.dbf file.

string is arbitrary and provides an indication of the source of the attribute. In practice, attributes belong to one of three (3) groups: CDB, Geomatics, and Vendor. If the extended-level attribute is one of the CDB attributes of section 5.7.1.2.7.5, the group name is “CDB”. If the extended-level attribute belongs to one of the Geomatics standards (such as “DIGEST”, “VMAP”, “SEDRIS”, “DGIWG”, “UHRB”), it is recommended to use the name of the standard as the group name. If the extended-level attribute is a vendor-specific attribute, then the group name should represent the name of the vendor (such as “CAE-M”, “Presagis”, “Thales”, “Rockwell”).

Column 3 – EAC (Environment Attribute Code): The third column provides a unique four-digit numeric code for each attribute type. The codes for the CDB attributes can be found in section 5.7.1.3, CDB Attributes. Note however, that the codes for the Geomatics and Vendor attributes are not specified by this Specification. Instead, this Specification provides a metadata schema that allows developers to describe these attributes. See section 5.1.7, CDB Attributes Metadata, for details.

Column 4 – EAV (Environment Attribute Value): The fourth column provides a data value for the attribute. The data value is represented by general-purpose 16-character alphanumeric string. In the case where more than 16-characters are needed to represent a data value, the remaining characters are provided by appending consecutive row(s) with the same GRP and EAC values; the value of LNK is incremented for each of the consecutive row(s). The interpretation of the data value is governed by metadata that describes the data type, the data format, the allowable range of the data, the numerical precision of the data, the units associated with the data, etc for each attribute type.

5.7.1.2.7.5 Example

The following example illustrates the relation between the vector data and attribute files where the instance, class, and extended-level attributes are stored. The example focuses on extended-level attributes. Note that it is possible to extend the list of instance and class attributes through the use of the CEAI, GEAI, and VEAI attributes.

The attributes associated with the instance of Shape 2 are extended with CDB attributes because CEAI has the value 4; that is an index into the Extended-level attributes dBase file, it points to record 4. By following the link (LNK) in each record, the complete list of extended attributes contains records 4, 5, and 8. These records add 3 CDB attributes: 5, 54, and 25. These codes respectively correspond to APID, RWID, and GAID. Their respective values are Airport CYUL, Runway 06L, and Gate B23.

The attributes that belong to the “Container” class are also extended with CDB attributes as indicated by the value 6 of the CEAI attribute. Record 6 adds CDB attribute 29, LACC, with a value of 1; record 7 adds CDB attribute 60, SSC, with a value of 84.

The attributes of the “Railroad” class are extended by Geomatics attributes as indicated by GEAI and its value of 1. This adds 3 DIGEST geomatics attributes (numbered 1, 2 and 3) that are defined in Geomatics_Attributes.xml.

Finally, the “Highway” class attributes are extended with a single vendor attribute stored in record 9 and 10 (VEAI points to record 9 which points to record 10). The client detects that this is a single attribute (and not two separate attributes) because the two records have identical values for their GRP and EAC attributes. The vendor is identified as “ABC Inc.”; attribute 1, defined in Vendor_Attributes.xml, has the value “1234567890ABCDEF GHIJ”.

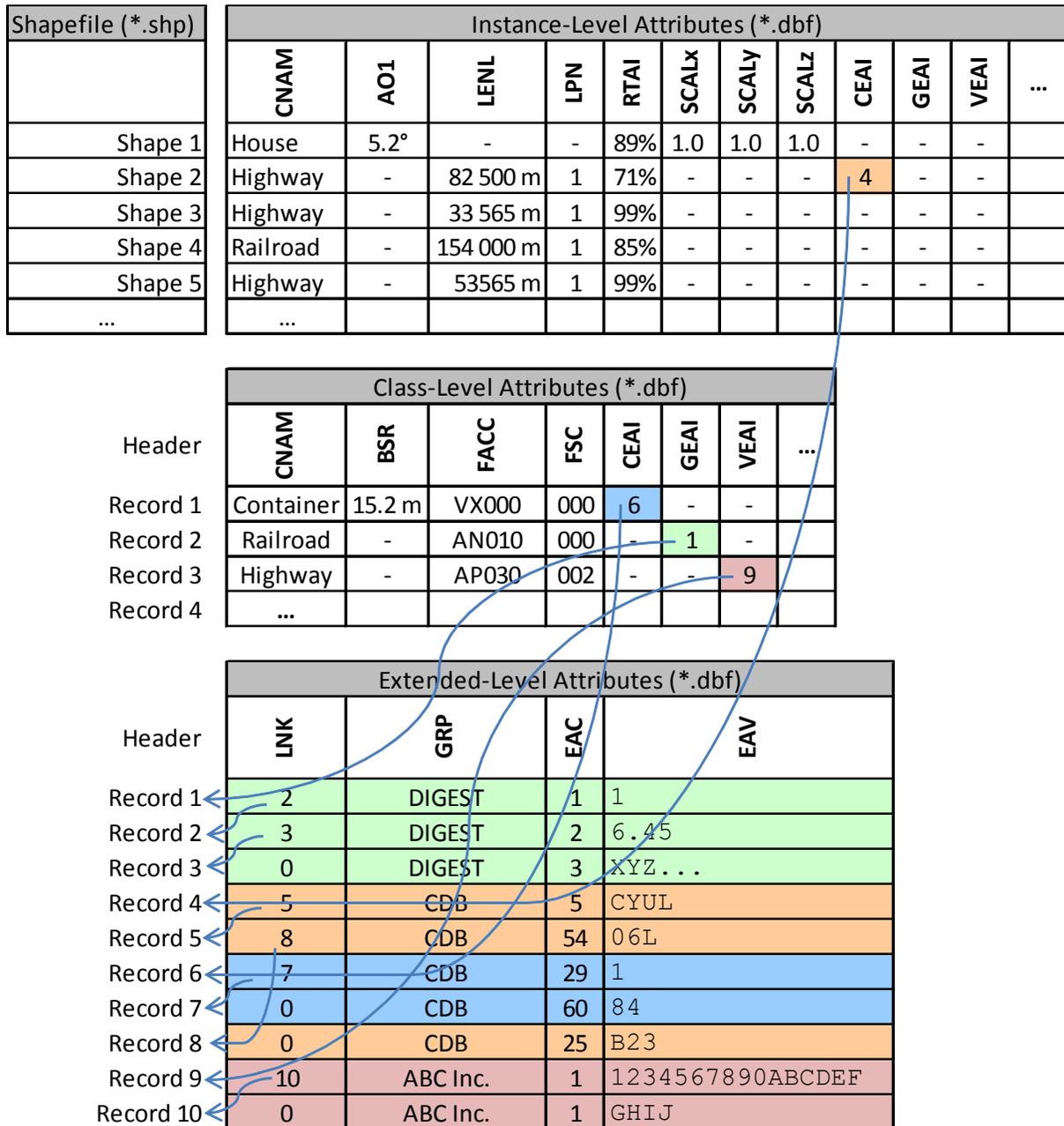


Figure 5-30: Relation between Shapes and Attributes

Note that it is possible to simultaneously extend a record (instance and class) with CDB, Geomatics, and vendor attributes. The example does not illustrate this situation to keep it (relatively) simple.

5.7.1.3 CDB Attributes

This section provides a list and description of the attributes that are governed by the CDB standard. Note that it is possible to provide attributes other than those listed here by making use of the Geomatics and Vendor Extended-level attribution schema.

5.7.1.3.1 *ATARS Extended Attribute Code (AEAC) – Deprecated*

Description: See OGC CDB Best Practice

5.7.1.3.2 *Absolute Height Flag (AHGT)*

Description: Indicates how to interpret the Z component of a vertex.

- Identifier: AHGT
- Code: 0002
- Data Type: Logical
- Length: 1 character
- Format: N/A
- Range: F, f, N, n (false) and T, t, Y, y (true)
- Usage Note: Optional. Specifies how to interpret vector data type features with a Z component. If AHGT is true, the feature is positioned to the value specified by the Z component (Absolute Terrain Altitude), irrelevant of the terrain elevation dataset. If AHGT is false or not present, the feature is positioned to the value specified by the underlying terrain offset by the Z component value. Refer to section 5.7.1.1, Vector Type Usage and Conventions for more details. AHGT can be present only in datasets using PointZ, PolylineZ, PolygonZ and MultiPointZ vector data types types. AHGT should not be present for all other vector data types or must be ignored otherwise. Refer to Section 6.3 in Volume 7: CDB Data Model Guidance (formerly Appendix A) – “How to Interpret the AHGT, HGT, BSR, BBH, and Z Attributes” for additional usage guidelines.
- Unit: N/A
- Default: False
- Compatibility: OGC CDB 1.0

NOTE: It is recommended that the AHGT flag be set to false because it facilitates the creation of CDB compliant datasets that are independent of each other. When the Z coordinate (altitude) of a feature is relative to the ground, the terrain elevation dataset can be updated without the need to re-compute the altitude of the feature.

CAUTION: When the AHGT flag is set to true, the feature will be at a fixed WGS-84 elevation independently of the terrain LOD selected by the client-device. As a result, there is no guarantee that the feature (and its modeled representation) will remain above the terrain across all terrain LODs.

RECOMMENDATION: Limit the use of AHGT=TRUE to data whose source is inherently absolute. Such source data include geodetic marks or survey marks that provide a known position in terms of latitude, longitude, and altitude. Good examples of such markers are boundary markers between countries.

5.7.1.3.3 *Angle of Orientation (AO1)*

Description: Angle of Orientation with greater than 1 degree resolution – The angular distance measured from true north (0°) clockwise to the major (Y) axis of the feature.

- Identifier: AO1
- Code: 0003
- Data Type: numeric
- Length: 7 characters
- Format: floating-point (recommended precision of 3.3)
- Range: 0.000 to 360.000
- Usage Note: Recommended. CDB readers should default to a value of 0.000 if AO1 is missing. Applicable to Point, Light Point, Moving Model Location and Figure Point features. When used in conjunction with the PowerLineNetwork dataset, AO1 corresponds to the orientation of the Y-axis of the modeled pylon. The modeled pylon should be oriented (in its local Cartesian space) so that the wires nominally attach along the Y-axis. Refer to Section 6.1 Volume 7: CDB Data Model Guidance (formerly Appendix A) – “Creating a 3D Model for a Powerline Pylon” for additional usage guidelines.
- Unit: degree
- Default: 0.000
- Compatibility: OGC CDB 1.0, DIGEST v2.1

5.7.1.3.4 *Airport Feature Name (APFN) – Deprecated*

Description: See OGC CDB Best Practice

5.7.1.3.5 *Airport ID (APID)*

Description: A unique alphanumeric identifier that points to a record in the NavData Airport or Heliport dataset (i.e., a link to the Airport or the Heliport description in the NavData dataset). This ID is the value of the field Ident of the Airport or Heliport dataset. Note that all of the lights located in vector datasets that are associated with the operation of an airport (including runway lights and lighting systems) are required to reference an airport or heliport in the NavData

dataset. All man-made features associated with an airport or heliport must be assigned an APID attribute; the APID attribute is not required for features unrelated to airports or heliports.

- Identifier: APID
- Code: 0005
- Data Type: alphanumeric
- Length: 6 characters
- Format: N/A
- Range: N/A
- Usage Note: Recommended for all Airport Light Points and airport-related T2DModels (such as runway/taxiway/apron surfaces, and markings) Failure to appropriately tag airport culture with APID attribute will result in reduced control of airport-related culture by simulator. Optional for Location Points, Environmental Light Points, and Moving Model Location features that fall within the confines of an airport and for which control of the feature is desirable.
- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0

5.7.1.3.6 Bounding Box Height (BBH)

5.7.1.3.7 Bounding Box Width (BBW)

5.7.1.3.8 Bounding Box Length (BBL)

Description: The Height/Width/Length of the Bounding Box of the 3D model associated with a point feature. It is the dimension of the box centered at the model origin and that bounds the portion of the model above its XY plane, including the envelopes of all articulated parts. BBH refers to height of the box above the XY plane of the model, BBW refers to the width of the box along the X-axis, and BBL refers to the length of the box along the Y-axis. Note that for 3D models used as cultural features, the XY plane of the model corresponds to its ground reference plane. The value of BBH, BBW and BBL should be accounted for by client-devices (in combination with other information) to determine the appropriate distance at which the model should be paged-in, rendered or processed. BBH, BBW and BBL are usually generated through database authoring tool automation.

- Identifiers: BBH, BBW, BBL
- Codes: 0006, 0007, 0008
- Data Type: numeric
- Length: 9 characters
- Format: floating-point (recommended precision 5.3)
- Range: 0.000 to 99999.999
- Usage Note: Optional on features for which a MODL has been assigned. The dimension of the bounding box is intrinsic to the model and identical for all LOD representations. Refer to Section 6.3 Volume 7: CDB Data Model Guidance (formerly

Appendix A) – “How to Interpret the AHGT, HGT, BSR, BBH, and Z Attributes” for additional usage guidelines.

- Unit: meters
- Default: BBH defaults to the value of BSR
 - BBW and BBL default to twice the value of BSR
- Compatibility: OGC CDB 1.0

5.7.1.3.9 *Boundary Type (BOTY)*

Description: A value that uniquely attributes a boundary according to the enumerators found here.

- Identifier: BOTY
- Code: 0009
- Data Type: Enumeration per Table 5-19: Boundary Type Enumeration Values
- Length: 3 characters
- Format: integer
- Range: 0 to 999
- Usage Note: Optional. See table below for a list of accepted values. Can be used only with Boundary Point, Linear or Polygon Feature Datasets (which are part of the Geopolitical Datasets)
- Unit: N/A
- Default: 0
- Compatibility: OGC CDB 1.0

Table 5-19: Boundary Type Enumeration Values

BOTY Code	Description
0	Unknown
1	Continental
2	International
3	Interstate
4	Inter-provincial
5	Territorial
6	Economic
7	Regional
8	Communal
9	Tourist
10	Private Zone
11	Military District
12	Disputed
13	Populated Place

14	Non-capital City
15	Time Zone Delimiter
16	International Date Line
17	Capital City
997	Unpopulated
998	Not Applicable
999	Other

5.7.1.3.10 *Bounding Sphere Radius (BSR)*

Description: The radius of a feature. In the case where a feature references an associated 3D model, it is the radius of the hemisphere centered at the model origin and that bounds the portion of the model above its XY plane, including the envelopes of all articulated parts. Note that for 3D models used as cultural features, the XY plane of the model corresponds to its ground reference plane. The value of BSR should be accounted for by client-devices (in combination with other information) to determine the appropriate distance at which the model should be paged-in, rendered or processed. When the feature does not reference a 3D model, BSR is the radius of the abstract point representing the feature (e.g., a city).

- Identifier: BSR
- Code: 0010
- Data Type: numeric
- Length: 9 characters
- Format: floating-point (recommended precision 5.3)
- Range: 0.000 to 99,999.999
- Usage Note: Mandatory for features for which a MODL has been assigned, but optional for geopolitical point features. The dimension of the bounding sphere is intrinsic to the model and identical for all LOD representations. Refer to Section 6.3 Volume 7: CDB Data Model Guidance (formerly Appendix A) – “How to Interpret the AHGT, HGT, BSR, BBH, and Z Attributes” for additional usage guidelines.
- Unit: meters
- Default: None
- Compatibility: OGC CDB 1.0

5.7.1.3.11 *CDB Extended Attribute Index (CEAI)*

Description: An index that points to a row entry of a CDB Extended Attribution file for the current dataset. This entry permits users to store an index to a link list set of CDB-specific attributes. CDB-compliant devices must be capable of reading and interpreting this field. Usage of this attribution is not portable to other simulators because it falls outside of the documented CDB attribution scheme. The CDB Extended Attribution file should be located in the same directory as the instance-level attribution file. An empty CEAI attribute is allowed. Note that the first entry in the CDB Extended Attribution file has an index of 1.

- Identifier: CEAI

- Code: 0011
- Data Type: numeric
- Length: 6 characters
- Format: integer
- Range: 1 to 999,999
- Usage Note: Optional. Use when CDB extended attribution is required. A “blank” or a value of 0 indicates that there are no CDB Extended attributes.
- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0.

5.7.1.3.12 CDB Extended Attribute Code (CEAC) – Deprecated

Description: See OGC CDB Best Practice.

5.7.1.3.13 Composite Material Index (CMIX)

Description: Index into the Composite Material Table is used to determine the Base Materials composition of the associated feature. Refer to Section 2.5, Material Naming Conventions for a description on material naming conventions.

- Identifier: CMIX
- Code: 0013
- Data Type: numeric
- Length: 6 characters
- Format: integer
- Range: 0 to 999,999
- Usage Note: Mandatory. Recommended to specify materials on polygons of Moving Models.
- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0

5.7.1.3.14 Class Name (CNAM)

Description: A name that represents the Attribution Class. The class-level attribution schema is described in Section 5.7.1.2.7.2, Class-level Schema. Attributes are referenced via this classname. The classname is used as the primary key to perform searches within the Dataset Class Attribute file.

- Identifier: CNAM
- Code: 0014
- Data Type: text
- Length: 32 characters
- Format: lexical
- Range: N/A

- Usage Note: Each row of a class-level database file or table must have a valid CNAM entry; the CNAM must be unique within the file. Each row of an instance-level *.dbf can optionally use the CNAM to refer to class attributes; blank indicates “no class attribute”.
- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0

5.7.1.3.15 *Damage Level (DAMA)*

Description: Represents the level of damage of the feature and its model, if applicable. The level is expressed as a percentage where a value of 0 means no damage at all and a value of 100 means fully damaged and completely destroyed. In the case of network datasets, the level of damage should be interpreted as a measure of the incapacity of the feature to perform its function. For instance, a road network whose damage level is 75% tells the client that it is only able to perform 25% of its intended function. As a result, a certain client may decide that it cannot use the road network while another client may continue to do so.

- Identifier: DAMA
- Code: 0067
- Data Type: numeric
- Length: 3 characters
- Format: integer
- Range: 0 to 100
- Unit: Percentage
- Default: 0
- Compatibility: OGC CDB 1.0
- Usage Note: In the context of HLA/DIS, the concept of DAMA maps directly to the concepts of Damage State for which the standards define 4 states named No Damage, Slight Damage, Moderate Damage, and Destroyed. The CDB standard suggests the following mapping between CDB DAMA and HLA/DIS states.

From CDB to HLA/DIS	
DAMA < 25	No Damage
25 ≤ DAMA < 50	Slight Damage
50 ≤ DAMA < 75	Moderate Damage
75 ≤ DAMA	Destroyed

From HLA/DIS to CDB	
No Damage	0
Slight Damage	33
Moderate Damage	66
Destroyed	100

5.7.1.3.16 *DIGEST Extended Attribute Code (DEAC) – Deprecated*

Description: See OGC CDB Best Practice

5.7.1.3.17 *Depth below Surface Level (DEP)*

Description: The depth of a feature. If the feature has no modeled representation, its depth is measured as the distance from the surface level to the lowest point of the feature below the surface⁴⁰. If the feature has an associated 3D model, the depth is measured as the distance from the XY plane of the model to the lowest point of the model below that plane. DEP values are positive numbers.

- Identifier: DEP
- Code: 0016
- Data Type: numeric
- Length: 9 characters
- Format: floating-point (recommended precision 5.3)
- Range: 0.000 to 99999.999
- Usage Note: In the case of ground features, DEP refers to the portion of the feature (or its modeled representation) that is underground. In the case of moving models that are used as geotypical features, DEP refers to the portion of the model that is below the waterline (i.e., the XY plane). In the case of network linear features such as roads, railroads and powerlines, DEP refers to the depth of the feature under the ground in its vicinity. In the case of hydrographic features, DEP refers to the depth of rivers, lakes, etc⁴¹. This data is typically used by client-devices that need to determine whether or not a waterway is navigable by ships with a specific draw.
- Unit: meters
- Default: 0.000
- Compatibility: OGC CDB 1.0

5.7.1.3.18 *Directivity (DIR)*

Description: The side or sides of a feature that has the greatest reflectivity potential. This data is typically needed for Radar simulation. DIR is used solely for linear features in accordance to DFAD conventions. If DIR is not equal to 3, then AO1 is the angular distance measured from true north (0 deg) clockwise to the reflective side of the feature.

- Identifier: DIR
- Code: 0017
- Data Type: numeric
- Length: 3 characters.
- Format: integer. Enumerated per DIGEST
 - 1: Uni-directional
- 2: Bi-directional
 - 3: Omni-directional

⁴⁰ Surface here refers to the terrain in the immediate vicinity of the feature.

⁴¹ Note, that the CDB has provision for a raster dataset to represent the bathymetry. When provided, the dataset provides a much more detailed underwater profile of hydrographic features.

- Range: 0 to 999
- Usage Note: Recommended for linear features. If absent, client-devices are required to default to a value of 3 – Omni-directional
- Unit: N/A
- Default: 3
- Compatibility: OGC CDB 1.0 and DIGEST

5.7.1.3.19 *Density Measure (DML)*

Description: Percentage light coverage at night (expressed as a percentage) within the area delimited by an polygon feature.

- Identifier: DML
- Code: 0018
- Data Type: numeric
- Length: 3 characters
- Format: integer
- Range: 0 to 100
- Usage Note: Recommended. Applies to Geopolitical Dataset polygon features that delineate inhabited areas. If this field is absent, client-devices shall assume 0%.
- Unit: Percentage
- Default: 0
- Compatibility: OGC CDB 1.0

5.7.1.3.20 *Density Measure (% roof cover) (DMR)*

Description: Roof cover measure by percent within area of feature.

- Identifier: DMR
- Code: 0019
- Data Type: numeric
- Length: 3 characters
- Format: integer
- Range: 0 to 100
- Usage Note: Recommended for Polygon features. If absent, client-devices shall assume 0%.
- Unit: percentage
- Default: 0
- Compatibility: OGC CDB 1.0, DIGEST 2.1

5.7.1.3.21 *Density Measure (structure count) (DMS)*

Description: Number of man-made, habitable structures per square kilometer.

- Identifier: DMS
- Code: 0020
- Data Type: numeric

- Length: 5 characters
- Format: integer
- Range: 0 to 99,999 (Note: differs from DIGEST range of -32767 to 32768)
- Usage Note: Recommended for Polygon features. If absent, client-devices shall assume 0.
- Unit: N/A
- Default: 0
- Compatibility: OGC CDB 1.0, DIGEST 2.1

5.7.1.3.22 Density Measure (% tree/canopy cover) (DMT)

Description: Canopy cover measure by percent within area of feature during the summer season.

- Identifier: DMT
- Code: 0021
- Data Type: numeric
- Length: 3 characters
- Format: integer
- Range: 0 to 100
- Usage Note: Recommended for Polygon features. If absent, client-devices shall assume 0%.
- Unit: percentage
- Default: 0
- Compatibility: OGC CDB 1.0, DIGEST 2.1

5.7.1.3.23 End Junction ID (EJID)

Description: A Junction Identification Number that is used to virtually connect the end point of a lineal to another point, lineal or polygon feature. Lineal features stored in the same vector file having the same SJID or EJID are connected. Lineal features stored in different vector files having the same SJID or EJID as the JID listed in the corresponding tile 2D relationship file are connected.

- Identifier: EJID
- Code: 0022
- Data Type: text numerals
- Length: 20 characters
- Format: unsigned integer64 as character string
- Range: 0 to $(2^{64} - 1)$
- Usage Note: Mandatory for all features belonging to Topological Network Datasets. Attribute is stored as a character string representing an unsigned 64-bit number, and requires conversion back into numerical representation by client reader. This is due to the 32-bit limitation on integer values within dBASE files.
- Unit: N/A
- Default: None

- Compatibility: OGC CDB 1.0

5.7.1.3.24 Feature Attribute Classification Code

Description: This code used to distinguish and categorize features within a dataset. The enumerated codes are listed in /CDB/Metadata/Feature_Data_Dictionary.xml.

- Identifier: FACC
- Code: 0023
- Data Type: text
- Length: 5 characters
- Format: two alpha characters following by three digits
- Range: N/A
- Usage Note: Mandatory
- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0, DIGEST 2.1

5.7.1.3.25 Feature Sub Code (FSC)

Description: This code, in conjunction with the feature code is used to distinguish and categorize features within a dataset. The enumerated codes are in accordance to Section 1.5.

- Identifier: FSC
- Code: 0024
- Data Type: numeric
- Length: 3 characters
- Format: integer. Enumerated per Section 1.5
- Range: 0 to 999
- Usage Note: Mandatory
- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0

5.7.1.3.26 Gate ID (GAID)

Description: A unique alphanumeric identifier (for the airport in question) that is consistent with the IDENT attribute name within the NavData Gate dataset. This ID is the value of the Gate Identifier of the Gate dataset and can be used to extract additional information such as the gate position and bearing.

- Identifier: GAID
- Code: 0025
- Data Type: alphanumeric
- Length: 6 characters
- Format: N/A
- Range: N/A

- Usage Note: Recommended and Optional usages are per Table 5-27: Allocation of CDB Attributes to Vector Datasets. Typically used (but not limited to) for models such as docking systems, marshallers and other models that are logically associated with a Terminal gate and that require some level of control by the simulation application.
- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0

5.7.1.3.27 Geomatics Extended Attribute Index (GEAI)

Description: An index that points to a row entry of a Geomatics Extended Attribution file for the current dataset. This entry permits users to store an index to a link list set of Geomatics-specific attributes. CDB-compliant devices are not mandated to read and interpret this field. Usage of this attribution is not portable to other simulators because it falls outside of the documented CDB attribution scheme. The Geomatics Extended Attribution file should be located in the same directory as the instance-level attribution file. An empty GEAI attribute is allowed. Note that the first entry in the Geomatics Extended Attribution file has an index of 1.

- Identifier: GEAI
- Code: 0026
- Data Type: numeric
- Length: 6 characters
- Format: integer
- Range: 1 to 999,999
- Usage Note: Optional. Use when Geomatics extended attribution is required. A “blank” or a value of 0 indicates that there are no Geomatics Extended attributes.
- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0.

5.7.1.3.28 Height above Surface Level (HGT)

Description: The height of a feature. If the feature has no modeled representation, its height is measured as the distance from the surface level (ground or water) to the tallest point of the feature above the surface⁴². If the feature has an associated 3D model, the height is measured as the distance from the XY plane of the model to the highest point of the model above that plane. HGT values are positive numbers.

- Identifier: HGT
- Code: 0027
- Data Type: numeric
- Length: 7 characters
- Format: floating-point (recommended precision 4.2)
- Range: 0.00 to 9999.99

⁴² Surface here refers to the terrain in the immediate vicinity of the feature.

- Usage Note: In the case of ground features, HGT refers to the portion of the feature (or its modeled representation) that is meant to be above ground. In the case of network lineal and polygon features such as roads, railroads, powerlines, or forest, HGT refers to the elevation of the feature relative to the terrain in its immediate vicinity.
- Unit: meters
- Default: 0.00
- Compatibility: OGC CDB 1.0

5.7.1.3.29 Junction ID (JID)

Description: A Junction Identification Number that is used to virtually connect a point or a polygon feature to another point, linear or polygon feature. Features stored in the same vector file having the same JID are connected. Features stored in different vector files having the same JID as the JID listed in the corresponding tile 2D relationship file are connected. When JID is associated to a polygon feature, it necessarily connects to the first point of the polygon feature.

- Identifier: JID
- Code: 0028
- Data Type: text numerals
- Length: 20 characters
- Format: unsigned integer64 as character string
- Range: 0 to $(2^{64} - 1)$
- Usage Note: Mandatory for all features belonging to Topological Network Datasets. Attribute is used in 2D relationship file. Attribute is stored as a character string representing an unsigned 64-bit number, and requires conversion back into numerical representation by client reader. This is due to the 32-bit limitation on integer values within dBASE files.
- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0

5.7.1.3.30 Location Accuracy (LACC)

Description: A precision value used to quantify the relative precision of the Location point representing the specific GeoPolitical Location.

- Identifier: LACC
- Code: 0029
- Data Type: numeric
- Length: 3 characters
- Format: integer
- Range: 0 to 999
- Usage Note: Optional. See Table 5-20: Location Accuracy Enumeration Values for a list of accepted values.
- Unit: meters.
- Default: 0

- Compatibility: OGC CDB 1.0

Table 5-20: Location Accuracy Enumeration Values

LACC Code	Description
0	Unknown
1	Better or equal to 10 m.
2	Better or equal to 100 m.
3	Better or equal to 250 m.
4	Better or equal to 500 m.
5	Better or equal to 1200 m.
6	Greater than 1200 m.
997	Unpopulated
998	Not Applicable
999	Other

5.7.1.3.31 Length of Lineal (LENL)

Description: The length of a lineal. If the feature has been clipped to a tile boundary, the length still gives the initial full length of the object prior to the clipping operation, and if it belonged to a topological network, LENL will represent the distance between the two closest junction points encompassing this linear segment. Note the Length attribute is not used to define a bounding sphere associated to an object, but rather to provide a weight to the relative length of the lineal as compared to others.

- Identifier: LENL
- Code: 0030
- Data Type: numeric
- Length: 6 characters
- Format: integer
- Range: 0 to 999,999
- Usage Note: Mandatory for all networked lineal features. Length computation should be account for the earth's curvature.
- Unit: meters
- Default: None
- Compatibility: OGC CDB 1.0, SEDRIS (EA = 562)

5.7.1.3.32 Light Material Index (LMIX) – Deprecated

Description: Please see the OGC CDB Best Practice

5.7.1.3.33 Feature (or Location) Name (LNAM)

Description: A toponym – a general term for any place or geographical entity. The attribute is used to give a proper noun (a human readable name) to any feature from any vector dataset.

- Identifier: LNAM
- Code: 0032
- Data Type: text
- Length: 32 characters
- Format: lexical
- Range: N/A
- Usage Note: The use of LNAM goes from the name of a City to the name of a Road, to the name of a Building, etc. Multiple names are possible when using LNAM as an extended attribute. When more than one name is provided, they must appear in order from the shortest name to the longest one.
- Unit: N/A
- Default: Blank
- Compatibility: OGC CDB 1.0

5.7.1.3.34 *Location Type (LOTY)*

Description: A value that uniquely attributes a location feature according to the enumerators found here.

- Identifier: LOTY
- Code: 0033
- Data Type: Enumeration per Table 5-21: Location Type Enumeration Values
- Length: 3 characters
- Format: integer
- Range: 0 to 999
- Usage Note: Optional. Applicable to Geopolitical Dataset polygon features. See table below for a list of accepted values. Can be used only with Location Point, Lineal or Polygon Feature Datasets (which are part of the Geopolitical Datasets)
- Unit: In cases where the location represents a bounded area, the approximate geometric center is assumed.
- Default: 0
- Compatibility: OGC CDB 1.0.

Table 5-21: Location Type Enumeration Values

LOTY Code	Description
0	Unknown
1	Continent
2	Country
3	State
4	Capital
5	Province
6	City
7	Municipality
997	Unpopulated
998	Not Applicable
999	Other

5.7.1.3.35 Light Phase (LPH)

Description: Used for all light types that are periodic in nature (rotating, blink, flashing, etc). The value of LPH controls the phase of the light relative to all other lights that share the same LTYP. All other light characteristics, including frequency and duration are implicitly determined by the LTYP.

- Identifier: LPH
- Code: 0034
- Data Type: numeric
- Length: 3 characters
- Format: integer
- Range: 0 to 999
- Usage Note: Optional. In absence of a value, LPH defaults to a value of 0.
- Unit: thousands of a cycle
- Default: 000
- Compatibility: OGC CDB 1.0

5.7.1.3.36 Layer Priority Number (LPN)

Description: A priority number that establishes the relative priority of overlapping features. LPN establishes the order (starting from 0 for lowest priority) by which overlapping features are processed by client-devices.

- Identifier: LPN
- Code: 0035
- Data Type: numeric
- Length: 5 characters
- Format: integer
- Range: 0 to 32767

- Usage Note: Mandatory for terrain constraint features that overlap one another. For all other features, it is optional. LPN is derived from priority information stored and maintained by the authoring tools.
- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0

5.7.1.3.37 Lane/Track Number (LTN)

Description: The number of lanes on a road, tracks on railroad, or conductors on powerlines, including both directions.

- Identifier: LTN
- Code: 0036
- Data Type: numeric
- Length: 2 characters
- Format: integer
- Range: 0 to 99 (Note: differs from DIGEST range of -32767 to 32768)
- Usage Note: Recommended for Road, RailRoad, and PowerLine Network features. Optional for Hydrography Network features.
- Unit: N/A
- Default: 02 – for RoadNetwork lineal features
 - 01 – for RailRoadNetwork lineal features
 - 02 – for PowerLineNetwork lineal features
- Compatibility: CDB 3.0, DIGEST 2.1

5.7.1.3.38 Light Type (LTYP)

Description: A unique code corresponding to a Light Type; Annex J of this standard provides the supported light types. The light types follow a hierarchical organization provided by the light type naming conventions described in Section 2.3, Light Naming. The Lights.xml file establishes the correspondence between the LTYP code and the Light Type name.

- Identifier: LTYP
- Code: 0037
- Data Type: numeric
- Length: 4 characters
- Format: integer
- Range: 0 to 9999
- Usage Note: Mandatory for all Airport Light Point features, Environmental Light Point features.
- Unit: N/A
- Default: 0
- Compatibility: OGC CDB 1.0

5.7.1.3.39 Model Level Of Detail (MLOD)

Description: The level of detail of the 3D model associated with the point feature.

- Identifier: MLOD
- Code: 0038
- Data Type: numeric
- Length: 3 characters
- Format: integer
- Range: -10 to 23
- Usage Note: When used in conjunction with MODL, the MLOD attribute indicates the LOD where the corresponding MODL is found. In this case, the value of MLOD can never be larger than the LOD of the Vector Tile-LOD that contains it. When used in the context of Airport and Environmental Light Point features, the value of MLOD, if present, indicates that this lightpoint also exist in a 3D model found at the specified LOD. In such case, the value of MLOD is not constrained and can indicate any LOD.
- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0

5.7.1.3.40 Moving Model DIS Code (MMDC)

Description: A character string composed of the 7 fields of the DIS Entity Type.

- Identifier: MMDC
- Code: 0039
- Data Type: text
- Length: maximum of 29 characters
- Format: All seven fields of the DIS Entity Type separated by an underscore character (“_”): 1_2_3_4_5_6_7
- Range: N/A
- Usage Note: Mandatory for Moving Model Location features
- Unit: N/A
- Default: None
- Compatibility: CDB 3.0

5.7.1.3.41 Model Name (MODL)

Description: A string reference, the model name, which stands for the modeled geometry of a feature; in the case of buildings, this includes both its external shell and modeled interior.

- Identifier: MODL
- Code: 0040
- Data Type: text
- Length: 32 characters
- Format: per conventions described in Chapter 3, CDB Structure.
- Range: N/A

- Usage Note: Needed for Point features, Road Figure Point features, Railroad Figure Point features, Pipeline Figure Point features and Hydrography Figure Point features that are modeled using some industry format. MODL can also be used with Road Lineal features, Railroad Lineal features, Pipeline Lineal features and Hydrography Lineal and Polygon features. Note that it is not permitted to specify a value for MODL simultaneously with a value for MMDC.
- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0

5.7.1.3.42 Model Type (MODT)

Description: Indicates whether a feature is represented using a geotypical or geospecific model. Together, the MODT, FeatureCode, FSC, and MODL attributes identify a unique model into the CDB.

- Identifier: MODT
- Code: 0041
- Data Type: text
- Length: 1 character
- Format: "T" for geotypical, "S" for geospecific
- Range: N/A
- Usage Note: Needed for features that are modeled using a CDB specified format.
- Unit: N/A
- Default: "S"
- Compatibility: OGC CDB 1.0

5.7.1.3.43 Network Component Selector 1 (NCS1)

Description: Code that is used to identify the component selector 1 file which contain the point, linear, or polygon feature that is virtually connected.

- Identifier: NCS1
- Code: 0042
- Data Type: numeric
- Length: 4 characters
- Format: unsigned integer
- Range: 0 to 9999
- Usage Note: Mandatory for Network datasets. Attribute is used in 2D relationship file.
- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0

5.7.1.3.44 Network Component Selector 2 (NCS2)

Description: Code that is used to identify the component selector 2 file which contain the point, linear, or polygon feature that is virtually connected.

- Identifier: NCS2
- Code: 0043
- Data Type: numeric
- Length: 4 characters
- Format: unsigned integer
- Range: 0 to 9999
- Usage Note: Mandatory for Network datasets. Attribute is used in 2D relationship file.
- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0

5.7.1.3.45 Network Dataset Code (NDSC)

Description: Code that is used to identify the dataset code file which contain the point, lineal, or polygon feature that is virtually connected.

- Identifier: NDSC
- Code: 0044
- Data Type: numeric
- Length: 4 characters
- Format: unsigned integer
- Range: 0 to 9999
- Usage Note: Mandatory for Network datasets. Attribute is used in 2D relationship file.
- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0

5.7.1.3.46 Number of Instances (NIS) – Deprecated

Description: Please see OGC CDB Best Practice

5.7.1.3.47 Number of Indices (NIX) – Deprecated

Description: Please see OGC CDB Best Practice

5.7.1.3.48 Number of Normals (NNL) – Deprecated

Description: Please see OGC CDB Best Practice

5.7.1.3.49 Number of Texture Coordinates (NTC) – Deprecated

Description: Please see OGC CDB Best Practice

5.7.1.3.50 Number of Texel (NTX) – Deprecated

Description: Please see OGC CDB Best Practice

5.7.1.3.51 Number of Vertices (NVT) – Deprecated

Description: Please see OGC CDB Best Practice

5.7.1.3.52 *Population Density (POPD)*

Description: The number of inhabitants per square kilometer.

- Identifier: POPD
- Code: 0051
- Data Type: numeric
- Length: 5 characters
- Format: integer
- Range: 0 to 99999
- Usage Note: Applicable to Geopolitical features representing inhabited areas.
- Unit: Inhabitants per square kilometer
- Default: 0
- Compatibility: OGC CDB 1.0

5.7.1.3.53 *Populated Place Type (POPT)*

Description: A value that uniquely represents the Populated Place Attribution Type. This attribute should be used in conjunction with the BOTY attribute when BOTY has an (enumerator) value of 13 which corresponds to “Populated Place”

- Identifier: POPT
- Code: 0052
- Data Type: Enumeration per Table 5-22: Populated Place Type Enumeration Values
- Length: 3 characters
- Format: integer
- Range: 0 to 999
- Usage Note: Optional. Applies to Geopolitical Dataset polygon features that delineate inhabited areas. See table below for a list of accepted values.
- Unit: N/A
- Default: 0
- Compatibility: OGC CDB 1.0

Table 5-22: Populated Place Type Enumeration Values

POPT Code	Description
0	Unknown
1	Native Settlement
2	Shanty Town
3	Tent Dwellings
4	Inland Village
5	Small City (less than 20,000 inhabitants)
6	Medium City (between 20,000 and 500,000 inhabitants)
7	Large City (more than 500,000 inhabitants)

997	Unpopulated
998	Not Applicable
999	Other

5.7.1.3.54 *Relative Tactical Importance (RTAI)*

Description: Provides the Relative Tactical Importance of cultural features relative to other features for the purpose of client-device scene/load management⁴³. A value of 100% corresponds to the highest importance; a value of 0% corresponds to the lowest importance. When confronted with otherwise identical objects that differ only wrt to their Relative Tactical Importance, client-devices should always discard features with lower importance before those of higher importance in the course of performing their scene / load management function. As a result, a value of zero gives complete freedom to client-devices to discard the feature as soon as the load of the client-device is exceeded. The effectiveness of scene / load management functions can be severely hampered if large quantities of features are assigned the same Relative Tactical Importance by the modeler. In effect, if all features are assigned the same value, the client-devices have no means to distinguish tactically important objects from each other. Assigning a value of 1% to all objects is equivalent to assigning them all a value of 99%. Ideally, the assignment of tactical importance to features should be in accordance to a histogram similar to the one shown here. The shape of the curve is not critical, however the proportion of features tagged with a high importance compared to those with low importance is critical in achieving effective scene/load management schemes. It is illustrated here to show that few features should have an importance of 100 with progressively more features with lower importance. The assignment of the RTAI to each feature lends itself to database tools automation. For instance, RTAI could be based on a look-up function which factors the feature's type (FeatureCode or MMDC).

$$RTAI = f(FACC _or _MMDC)$$

The value of Relative Tactical Importance should be accounted for by client-devices (in combination with other information) to determine the appropriate distance at which the feature (and its modeled representation, if available) should be rendered or processed. Relative Tactical Importance is mandatory. It has no default value.

⁴³ Note that the importance of the model can be further modified at run-time at the simulator console through the scenario importance value assigned to the model.

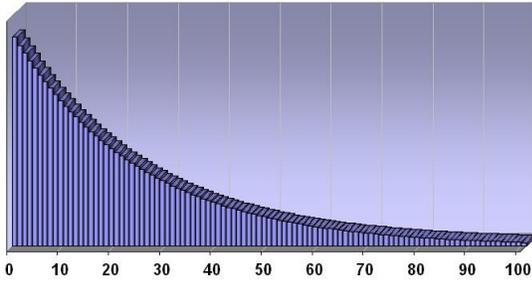


Figure 5-31: RTAI Typical Usage Histogram

- Identifier: RTAI
- Code: 0053
- Data Type: numeric
- Length: 3 characters
- Format: integer
- Range: 0 to 100
- Usage Note: Mandatory. All features should be tagged with an appropriate value for the reasons stated above.
- Unit: Percentage
- Default: None
- Compatibility: OGC CDB 1.0

5.7.1.3.55 Runway ID (RWID)

Description: An alphanumeric identifier that uniquely identifies a runway for a given airport; this ID must match the value of the field Ident of the Runway or Helipad dataset.

- Identifier: RWID
- Code: 0054
- Data Type: Alphanumeric
- Length: 6 characters
- Format: N/A
- Range: N/A
- Usage Note: Recommended for all Airport Light Points features. Failure to appropriately tag airport culture with RWID attribute will result in reduced control of runway-related (or helipad) culture by simulator. Optional for Point/Lineal/Polygon features, Location Points Features, Environmental Light Point features, and Moving Model Location features that are associated with a runway and for which control of the feature is desirable. The combination of RWID and APID points to a unique record of the NavData Runway or Helipad dataset components. Note that all of the lights and other features located in vector datasets that are associated with the operation of a runway or helipad are required to reference a runway or helipad in the NavData dataset; the RWID attribute is not required for features unrelated to a runway or helipad.
- Unit: N/A

- Default: None
- Compatibility: OGC CDB 1.0

5.7.1.3.56 *Scaling (SCALx)*

5.7.1.3.57 *Scaling (SCALy)*

5.7.1.3.58 *Scaling (SCALz)*

Description: A set of scaling factors, one of the model axis, to be applied to the rendering of model geometry by the client-device. A value of 1.0 instructs the client-devices to use the model as-is. The physical dimension of models processed by client-device should approach zero, as SCALing tends to zero. The value of SCALing should also be accounted for by client-devices (in combination with other information) to determine the appropriate distance at which the model should be paged-in, rendered or processed. All three SCALing factors are optional. Values of zero and negative values are not permitted.

- Identifiers: SCLAx, SCALy, SCALz
- Codes: 0055, 0056, 0057
- Data Type: numeric
- Length: 9 characters
- Format: floating-point (recommended precision 3.5)
- Range: 000.00001 to 999.99999
- Usage Note: Optional. CDB readers should default to a value of 1.0 if SCALx, SCALy, or SCALz is missing.
- Unit: N/A
- Default: 1.0
- Compatibility: CDB 3.0

5.7.1.3.59 *Start Junction ID (SJID)*

Description: A Junction Identification Number that is used to virtually connect the start point of a lineal to another point, lineal or polygon feature. Lineal features stored in the same vector data file having the same SJID or EJID are connected. Lineal features stored in different vector files having the same SJID or EJID as the JID listed in the corresponding tile 2D relationship file are connected.

- Identifier: SJID
- Code: 0058
- Data Type: text numerals
- Length: 20 characters
- Format: unsigned integer64 as character string
- Range: 0 to $(2^{64} - 1)$
- Usage Note: Mandatory for all features belonging to Topological Network Datasets. Attribute is stored as a character string representing an unsigned 64-bit number, and requires conversion back into numerical representation by client reader. This is due to the 32-bit limitation on integer values within dBASE files.

- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0

5.7.1.3.60 *Surface Roughness Description (SRD)*

Description: Describes the condition of the surface materials that may be used for mobility prediction, construction material, and landing sites.

- Identifier: SRD
- Code: 0059
- Data Type: Enumeration per Table 5-23: Surface Roughness Enumeration Values
- Length: 3 characters
- Format: integer
- Range: 0 to 999
- Usage Note: Recommended for Polygon features.
- Unit: N/A
- Default: 0
- Compatibility: OGC CDB 1.0

Table 5-23: Surface Roughness Enumeration Values

SRD Code	Description
0	Unknown
1	No surface roughness effect
2	Area of high landslide potential
3	Uncohesive surface material/flat
4	Rough
5	Angular
6	Rounded
11	Surface of numerous cobbles and boulders
12	Areas of stony terrain
13	Stony soil with surface rock
14	Stony soil with scattered boulders
15	Stony soil with numerous boulders
16	Numerous boulders
17	Numerous rock outcrops
18	Area of scattered boulders
19	Talus slope
20	Boulder Field
31	Highly fractured rock surface
32	Weathered lava flows

33	Unweathered lava flows
34	Stony soil with numerous rock outcrops
35	Irregular surface with deep fractures of foliation
36	Rugged terrain with numerous rock outcrops
37	Rugged bedrock surface
38	Sand dunes
39	Sand dunes/low
40	Sand dunes/high
41	Active sand dunes
42	Stabilized sand dunes
43	Highly distorted area, sharp rocky ridges
51	Stony soil cut by numerous gullies
52	Moderately dissected terrain
53	Moderately dissected terrain with scattered rock outcrops
54	Dissected floodplain
55	Highly dissected terrain
56	Area with deep erosional gullies
57	Steep, rugged, dissected terrain with narrow gullies
58	Karst areas of numerous sinkholes and solution valleys
59	Karst area of numerous sinkholes
60	Karst/hummocky terrain covered with large conical hills
61	Karst/hummocky terrain covered with low, broad-based mounds
62	Arroyo/wadi/wash
63	Playa/dry lake
64	Area of numerous meander scars and/or oxbow lakes
65	Solifluction lobes and frost scars
66	Hummocky ground, areas of frost heaving
67	Area of frost polygons
68	Area containing sabkhas
69	Area of numerous small lakes and ponds
70	Area of numerous crevasses
81	Area of numerous terraces
82	Quarries
83	Strip mines
84	Quarry/gravel pit
85	Quarry/sand pit
86	Mine tailings/waste piles
87	Salt evaporators

88	Area of numerous dikes
89	Area of numerous diked fields
90	Area of numerous fences
91	Area of numerous stone walls
92	Area of numerous man-made canals/drains/ditches
93	Area of numerous terraced fields
94	Parallel earthen mounds row crops
95	Area of numerous hedgerows
997	Unpopulated
998	Not Applicable
999	Other

5.7.1.3.61 Structure Shape Category (SSC)

Description: Describes the Geometric form, appearance, or configuration of the feature.

- Identifier: SSC
- Code: 0060
- Data Type: Enumeration per Table 5-24: Structure Shape Category Enumeration Values
- Length: 3 characters
- Format: integer
- Range: 0 to 999
- Usage Note: Recommended for Point features, and all Network Lineal/Polygon Figure Points features.
- Unit: N/A
- Default: 0
- Compatibility: OGC CDB 1.0, DIGEST 2.1

Table 5-24: Structure Shape Category Enumeration Values

SSC Code	Description
0	Unknown
1	Barrel, Ton
2	Blimp
3	Boat Hull (Float)
4	Bullet
5	<i>Reserved</i>
6	Conical/Peaked/NUN
7	Cylindrical (Upright)/CAN
9	<i>Reserved</i>

10	Pillar/Spindle
11	<i>Reserved</i>
12	Pyramid
13	<i>Reserved</i>
14	<i>Reserved</i>
15	Solid/filled
16	Spar
17	Spherical (Hemispherical)
18	Truss
19	With Radome
20	<i>Reserved</i>
21	Artificial Mountain
22	Crescent
23	Ferris Wheel
24	Enclosed
25	Roller Coaster
26	Lateral
27	Mounds
28	Ripple
29	Star
30	Transverse
31	<i>Reserved</i>
32	<i>Reserved</i>
33	<i>Reserved</i>
34	<i>Reserved</i>
36	Windmotor
38	<i>Reserved</i>
40	<i>Reserved</i>
46	Open
52	'A' Frame
53	'H' Frame
54	'T' Frame
56	'Y' Frame
57	<i>Reserved</i>
58	<i>Reserved</i>
59	Telescoping Gasholder (Gasometer)
60	Mast
61	Tripod

62	<i>Reserved</i>
63	<i>Reserved</i>
65	Cylindrical with flat top
66	Cylindrical with domed top
71	Cylindrical/Peaked
73	Superbuoy
74	'T' Frame
75	Tetrahedron
76	Funnel
77	Arch
78	Multi-Arch
79	Round
80	Rectangular
81	Dragons Teeth
82	I-Beam
83	Square
84	Irregular
85	Diamond Shaped Buoy
86	Oval
87	Dome
88	Spherical with Column Support
89	Cylindrical or Peaked with tower support
90	High-Rise Building
91	Cylindrical
92	Cubic
93	Pole
94	Board
95	Column (Pillar)
96	Plaque
97	Statue
98	Cross
107	Tower
108	Scanner
109	Obelisk
110	Radome, Tower Mounted
997	Unpopulated
998	Not Applicable
999	Other

5.7.1.3.62 Structure Shape of Roof (SSR)

Description: Describes the roof shape.

- Identifier: SSR
- Code: 0061
- Data Type: Enumeration per Table 5-25: Structure Shape of Roof Enumeration Values
- Length: 3 characters
- Format: integer
- Range: 0 to 999
- Usage Note: Recommended for Point features, and all Network Linear/Polygon Point Figures.
- Unit: N/A
- Default: 0
- Compatibility: OGC CDB 1.0, DIGEST 2.1

Table 5-25: Structure Shape of Roof Enumeration Values

SSR Code	Description
0	Unknown
6	Conical/Peaked/NUN
38	Curved/Round (Quonset)
40	Dome
41	Flat
42	Gable (Pitched)
43	<i>Reserved</i>
44	<i>Reserved</i>
45	<i>Reserved</i>
46	<i>Reserved</i>
47	Sawtooth
48	<i>Reserved</i>
49	<i>Reserved</i>
50	With Monitor
51	With Steeple
55	Flat with Monitor
58	<i>Reserved</i>
65	Gable with Monitor
65	<i>Reserved</i>
66	<i>Reserved</i>
71	<i>Reserved</i>
72	<i>Reserved</i>

77	With Cupola
78	With Turret
79	With Tower
80	With Minaret
997	Unpopulated
998	Not Applicable
999	Other

5.7.1.3.63 *Traffic Flow (TRF)*

Description: Encodes the general destination of traffic.

- Identifier: TRF
- Code: 0062
- Data Type: numeric
- Length: 3 characters
- Format: Integer. Enumerated per DIGEST 2.1. A few examples:
 - 3: One-way
 - 4: Two-way
- Range: 0 to 999
- Usage Note: Recommended on all Network Linear (except PowerLines) features.
- Unit: N/A
- Default: 003 – for RailRoadNetwork linear features
 - 004 – for other network lineal features
- Compatibility: OGC CDB 1.0, DIGEST 2.1

5.7.1.3.64 *Taxiway ID (TXID)*

Description: A unique alphanumeric identifier (for the airport in question).

- Identifier: TXID
- Code: 0063
- Data Type: alphanumeric
- Length: 6 characters
- Format: N/A
- Range: N/A
- Usage Note: Recommended usage and Optional usages are per table Table 5-27: Allocation of CDB Attributes to Vector Datasets. Failure to appropriately tag airport culture with TXID attribute will result in reduced control of taxiway-related culture by a simulation device.
- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0

5.7.1.3.65 *Urban Street Pattern (USP)*

Description: Describes the predominant geometric configuration of streets found within the delineated area of the feature.

- Identifier: USP
- Code: 0064
- Data Type: Enumeration per Table 5-26: Urban Street Pattern Enumeration Values
- Length: 3 characters
- Format: integer
- Range: 0 to 999
- Usage Note: Recommended for Polygon features.
- Unit: N/A
- Default: 0
- Compatibility: OGC CDB 1.0, DIGEST 2.1

Table 5-26: Urban Street Pattern Enumeration Values

USP Code	Description
0	Unknown
2	Rectangular/Grid-Regular
3	Rectangular/Grid-Irregular
4	Curvilinear (cluster)
6	Concentric / Radial-Regular
7	Concentric / Radial-Irregular
9	Mixed-Curvilinear (cluster) and Rectangular (grid)
10	Mixed-Concentric / Radial and Rectangular (grid)
11	Mixed-Curvilinear (cluster) and Concentric / Radial
12	<i>Reserved</i>
13	Linear Strip
997	Unpopulated
998	Not Applicable
999	Other

5.7.1.3.66 *Vendor Extended Attribute Index (VEAI)*

Description: An index that points to a row entry of a VendorExtended Attribution file for the current dataset. This entry permits users to store an index to a link list set of Vendor-specific attributes. CDB-compliant devices are not mandated to read and interpret this field. Usage of this attribution is not portable to other simulators because it falls outside of the documented CDB attribution scheme. The Vendor Extended Attribution file should be located in the same directory as the instance-level attribution file. An empty VEAU attribute is allowed. Note that the first entry in the Vendor Extended Attribution file has an index of 1.

- Identifier: VEAI
- Code: 0065
- Data Type: numeric
- Length: 6 characters
- Format: integer
- Range: 1 to 999,999
- Usage Note: Optional. Use when Vendor extended attribution is required. A “blank” or a value of 0 indicates that there are no Vendor Extended attributes.
- Unit: N/A
- Default: None
- Compatibility: OGC CDB 1.0.

5.7.1.3.67 *Width with Greater Than 1 meter Precision (WGP)*

Description: For lineal features (such as roads, railways, runways, taxiways), WGP is a measurement of the shorter of two linear axes. For a bridge, the width is the measurement perpendicular to the axis between the abutments.

- Identifier: WGP
- Code: 0066
- Data Type: numeric
- Length: 9 characters
- Format: floating-point (recommended precision 5.3)
- Range: 0.000 to 99,999.999
- Usage Note: Recommended on all Network Lineal (except PowerLines) features. When supplied for coastline, WGP corresponds to average width of coastline due to water height variations (such as tides) and wave action.
- Unit: meters
- Default: None
- Compatibility: OGC CDB 1.0, DIGEST 2.1

5.7.1.4 Explicitly Modeled Representations

5.7.1.4.1 *Referenced by Point Features*

A point feature (whose position and attributes are stored in a vector data set) can also refer to an explicitly modeled representation.

A feature can point to an explicitly modeled representation of that feature that is stored in either the GTModel library, the MModel library or alternately embedded inside a CDB tile. In order to specify the modeled representation, the modeler must properly attribute the feature via the MODL, MLOD, MMDC and MODT attributes in the vector dataset that contains the feature.

For Point features, the CDB currently supports two types of explicitly modeled representations:

- OpenFlight models (Volume 6)
- RCS models (Volume 5)

Natural vector features (such as trees, bushes) are usually represented by geotypical models. The majority of man-made features can also be geotypical in nature. For instance, power pylons, telephone poles, or residential houses can all be represented with generic models that are typical in appearance to the real-world objects they represent. The modeler need only resort to a geospecific model if the application requires a model with the unique shape, appearance and/or properties of the real-world object.

5.7.1.5 Implicitly Modeled Representations

An implicitly modeled representation is one that is defined completely by the supplied attribution of the Dataset in which it is contained. Examples of implicitly modeled features are light-points.

5.7.1.6 Handling of Topological Networks

The CDB standard provides several interconnected topological networks consisting of multiple datasets. Each network dataset can be made of separate point features and or a series of points connected together using lineal and polygon features. Each lineal feature has a start and end nodes, which correspond to intersections when connected to two or more other lineal features, or connections when connected to a polygon; the other intermediate points are used to accommodate deviation from a straight line. Typically, network datasets (such as roads, railroads) conform to the ground; consequently, when the optional AHGT attribute is present, its value is set to false. Each network dataset is stored as a distinct vector dataset⁴⁴.

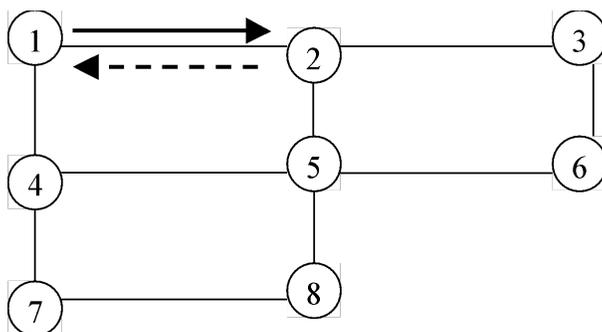


Figure 5-32: Example of a Topological Network

The CDB Topological networks are useful when one needs to determine the shortest path between two arbitrary nodes in the entire network; alternately, algorithms can use the network topology in combination with a “cost” parameter based on length (in the case of shortest path), traffic speed (in the case of fastest path), or some other criteria that can be derived from the attribution information associated with the network datasets.

The CDB Topological networks are used for the following purposes:

- To determine a route for features such as roads, rivers, railroads.
- To follow a route made of roads, rivers, railroads.

⁴⁴ Shapefile if Shapefiles are being used

- To avoid an obstacle; for example a tank may not be able to cross a river or be able to go over or under a pipeline.
- To efficiently process a “feature” for devices (such as radar) that do not require a high definition of the geometrical representation or do not need to represent more than one dimension.

The CDB Topological networks are optimized to facilitate road/river/railroad following tasks. They support the notion of directionality such as one-way roads or both ways for two-way roads, rivers. The vertex positions are physically positioned along the center of the feature’s longitudinal axis; for example, a road such as a dual-lane undivided highway, the vertex data lies along the stripes dividing the two lanes.

Features within the same or different network datasets are connected together using the junction identifier attributes SJID, EJID or JID. Two or more features having the same identifier values are considered virtually connected. This junction identifier allows, for instance, a primary road to connect to a secondary road, or a river to connect a lake (in same network datasets), or to connect a road and a river (in different network dataset).

Requirements Class - Vector datasets Topology (117-119)	
/req/core/tiled-vector-datasets-topology	
Target type	Operations
Dependency	Various XML schema
Requirement 117	/req/core/topology-attributes
Requirement 118	/req/core/topo-tile-clip
Requirement 119	/req/core/topo-jid-range

Requirement 117
http://www.opengis.net/spec/cdb/1.0/core/topology-attributes
For all topological network datasets the SJID, EJID or JID attributes <i>SHALL</i> be specified.

When not specified (i.e., blank), the feature is not connected to any other features. Volume 7: CDB Data Model Guidance (formerly Appendix A) provides guidelines on how to generate the junction identifiers.

Since the junction identifier is associated with a shape type feature, the following combinations are supported:

- Any point feature can be connected to any start or end point of a linear feature (point to linear connection), or to any start point of a polygon feature (point to polygon connection), using its JID attribute.
- Any start point of a linear feature can be connected to any point feature (point to linear connection), or to any start or end point of a linear feature (linear to linear connection), or to any start point of a polygon feature (linear to polygon connection), using its SJID attribute.
- Any end point of a linear feature can be connected to any point feature (point to linear connection), or to any start or end point of a linear feature (linear to linear connection), or to any start point of a polygon feature (linear to polygon connection), using its EJID attribute.
- Any start point of a polygon feature can be connected to any point feature (point to polygon connection), or to any start or end point of a linear feature (linear to polygon connection), using its JID attribute.

Connection information between two features located in two separate vector datasets⁴⁵ are explicitly listed in 2D relationship files. This standard currently specifies two types of 2D relationship files: the 2D relationship tile connection file which specifies connections of the same dataset feature between two adjacent tiles, and the 2D relationship dataset connection file which specifies connection of 2 or more different dataset and sub-dataset features within the same tile.

5.7.1.6.1 2D Relationship Tile Connection File

Requirement 118
http://www.opengis.net/spec/cdb/1.0/core/topo-tile-clip
The CDB Topological network is broken into tiles and therefore <i>SHALL</i> be clipped against tile boundaries. To ensure the connectivity between tile boundaries of a lineal feature, the resulting clipping point <i>SHALL</i> share the same junction identifier (JID) in both tiles

⁴⁵ Shapefiles if Shapefiles are being used.

This clipping point potentially exists in several Tile-LODs having a common boundary; in which case, all points representing the same clipping point share the same JID. Doing so ensures that connectivity between geocells and tiles is preserved. A clipping point can be identified by the application by checking the 2D relationship tile connection file. There is a 2D relationship tile connection file per network dataset tile. When the file is missing, it indicates there is no clipping point for the lineals belonging to the tile. The 2D relationship file is a dBASE file that contains a list of records made of 2 attributes; the Junction ID (JID) that identifies the start or end point of the clipped linear and the Network Component Selector 1 (NCS1) that identifies the network dataset lineal file. The dataset code file is implicit to the network dataset tile directory and the Network Component Selector 2 always represents a linear feature vector features (code 003) thus do not require to be included in the record. The coordinate of the tile adjacent to a clipping point can be determined using the latitude and longitude of that point.

If a connection between two linear features happens to be located exactly at a tile boundary, the lineal is obviously not clipped but a junction ID is allocated and included in the 2D relationship tile connection file.

In a 2D relationship tile connection file, no two records are identical; however JIDs may appear more than once with different NCS1, indicating a connection between network subdatasets.

5.7.1.6.2 2D Relationship Dataset Connection File

The CDB Topological network is made of several network datasets that in turn are made of several vector files (such as ShapeFiles). By specifying a junction identifier per feature, any features in any of these several vector files can in theory be connected to any other features located in a separate files (such as Shapefiles). A connection within a tile, which includes the tile boundaries, can be identified by the application by checking the 2D relationship dataset connection file. There is a 2D relationship dataset connection file per network dataset tile. This file contains all the connections between the sub-datasets of the corresponding network dataset with other network datasets. When the file is missing, it indicates there are no connections within the tile. The 2D relationship file is a database or other file that contains a list of records made of 4 attributes; the Junction ID (JID) that identifies the connected point, lineal or polygon features, the Network Dataset Code (NDSC) that identifies the network dataset the feature belongs to, and the Network Component Selectors (NCS1 and NCS2) that identify the network sub-dataset and the shape type. The tile coordinate and tile LOD is implicit to the Network Dataset tile directory.

All the records in the 2D relationship file are sorted per ascending JID. This has two advantages; it speeds up the search process when looking for a specific JID and it groups all the features that are connected together. In effect, there is always a minimum of two consecutive records with the same JID; the record belonging to the corresponding file dataset (or sub-dataset) and the record identifying the feature it connects to. Note that when a 2D relationship file specifies a connection to a feature belonging to different network dataset, the corresponding LOD file of this dataset may or may not exist. If the corresponding LOD file of the target dataset is missing, the application must look for the feature in the coarser LOD file of the target dataset. If the corresponding LOD file of the target dataset exists, the feature may be missing because it has

been removed by the off-line tool decimation process; when this is the case, the application must look for the feature in the finer LOD file of the target dataset.

5.7.1.6.3 Junction Identifier (SJID, EJID, and JID) Range

Requirement 119
http://www.opengis.net/spec/cdb/1.0/core/topo-jid-range
This version of the standard imposes that SJID, EJID and JID <i>SHALL</i> have unique values for all network datasets within the same geocell.

With a 64-bit range, it is practically impossible to run out of ID numbers. In the process of creating the unique identifier, special care must be taken by the off line tool in order to avoid duplicating the identifier at the geocell boundary for the clipping points when the modified or added features overlap two or more geocells.

Table 3-27, CDB LOD versus Feature Density, specifies the maximum number of elements in a tile for vector datasets. This maximum number is not affected by the number of added clipping point in a lineal feature. Although, this appears to lead to an unbounded number of points in a file, it is clamped to the geocell size. In practice, for a relative constant density, the number of clipping points diminishes as the LOD number increases.

5.7.1.7 Handling of Light Points

All of the information that is needed to instantiate the light point features is organized in accordance to the CDB terrain tile structure. Each instance of a light point feature refers to a light type defined by the CDB Standard via its shape attribution (LTYP). As a result, the entire definition of the light (i.e., its location, orientation and attributions) is self-contained within the vector data files.

The CDB standard defines a collection of CDB Light Types that includes airport/runway lighting systems, cultural lights, aircraft refueling systems, etc. The light types currently supported by the CDB standard are listed in Annex J of this standard; they are also listed in Lights.xml as specified in Section 2.3, Light Naming. While the CDB standard provides a rigorous definition for each type of light, its representation is entirely determined by the fidelity and the capabilities of client-devices.

5.7.1.8 Allocation of CDB Attributes To Vector Datasets

The CDB standard limits the applicability of each of the CDB attributes to certain vector datasets. This approach helps to reduce the number of columns (hence to reduce the size) of the dBase instance and class-level attribution files.

The allocation of CDB attributes to each of the Vector datasets is prescribed by Table 5-27 below.

Table 5-27: Allocation of CDB Attributes to Vector Dataset

5.7.2 Tiled Navigation Dataset

As described in section 5.2, the global navigation dataset is complemented by a tiled-based dataset of basic navigation information that refers to the corresponding geocell. Those are found in the \401_Navigation dataset which is subdivided into several components as listed in the next table.

Table 5-28: Tiled Navigation Dataset

CS1	CS2	File Extension	Component Name	Component Description
Dataset 401, Navigation				
001-046	001	Dependent on format or encoding	Tiled Navigation Dataset	Contains the basic Navigation records

Refer to Table below, List of Navigation Components, for a complete description of the possible values of CS1.

List of Navigation Components

Component Name	CS1	Vector Type	Component Description
Airport	1	Point	Area or land that is used (or intended for use) for the landing and take-off of aircraft.
AirRefueling	2	Point	A specifically designated airspace where air-to-air refueling operations are normally conducted.
AirRefuelingControl	3	Point	Information regarding the Air Traffic Control Center that controls the airspace within which the refueling track or anchor is located.
AirRefuelingFootnote	4	Point	Supplemental notes defining an Air Refueling component
AirRefuelingPoint	5	Point	Single Point from an Air Refueling structure
AirRefuelingSegment	6	LineString	Segment from an Air Refueling structure
AirspaceBoundary	7	Point	Designated airspace within which some or all aircraft may be subject to air traffic control.
AirwayRestriction	8	Point	Altitude and time restrictions for airways, airway segments, or sequences of airway segments
Approach	9	LineString	Preplanned instrument flight rule (IFR) for air traffic control approach procedures.
ArrestingGear	10	Point	Safety device consisting of engaging or catching devices, and energy absorption devices for the purpose of arresting both

			tail hook and/or non-tail hook equipped aircraft
COMMS	11	Point	Voice, radio communications, and facility call sign and frequencies available for same operations between the airport environment and aircraft.
ControlAirspace	12	Multipoint	Sequential listing of vertical and lateral limits, defining airspaces of different classifications, within which air traffic control service is provided
EnrouteAirway	13	Point	A specified route designed for channeling the flow of traffic as necessary for the provision of air traffic services
FirUir	14	Polygon	Flight Information region - Upper Information Region. Designated airspace within which some or all aircraft may be subject to air traffic control.
Gate	15	Point	Passenger gate at an airport
GLS	16	Point	GNSS Landing System
Helipad	17	Line	Designated area usually with a prepared surface used for take-off and landing of helicopters
Heliport	18	Point	Area or land intended to be used for landing and takeoff of helicopters
HoldingPattern	19	Point	Flight path maintained by an aircraft that is awaiting permission to land
ILS	20	Multipoint	Instrument landing system - Precision instrument approach system normally consisting of electronic components and visual aids
Marker	21	Point	Transmitter that radiates vertically a distinctive pattern for providing position information to aircrafts
MilitaryTrainingRoute	22	Point	Routes used by the Department of Defense and associated Reserve and Air Guard Units for the purpose of conducting low altitude navigation and tactical training in both IFR and VFR weather conditions below 10,000 feet MSL at airspeeds in excess of 250 KTS IAS.
MilitaryTrainingRouteAirspace	23	Point	Special use airspace or military operations area associated with a Military Training Route
MilitaryTrainingRouteDescription	24	Point	Supplemental information regarding a Military Training Route
MilitaryTrainingRouteOverlay	25	LineString	The width left and right of centerline based on a set of widths at Point Ident

			and another set of width at the Next Point Ident in one segment record.
MLS	26	Multipoint	Microwave Landing System - precision instrument approach system normally consisting of electronic components and visual aids
MSA	27	Point	Minimum Safe Altitude - altitude below which it is hazardous to fly owing to presence of high ground or other obstacles
Navaid	28	Multipoint	Electronic device on the surface, which provides point-to-point guidance information or position data to aircraft in flight
OffRouteTerrainClrAltitude	29	Polygon	Off-Route Terrain Clearance Altitude - Clearance altitudes in non-mountainous and in mountainous areas
ParachuteJumpArea	30	Point	An area designated for parachute jumping activities.
ParachuteJumpAreaBoundary	31	Polygon	Boundary of a Parachute Jump Area
PathPoint	32	Point	
PreferredRoute	33	Point	A system of routes designed to minimize route changes during the operational phase of flight and to aid in the efficient management of air traffic.
PresetSite	34	Point	Preset Site
RestrictiveAirspace	35	Polygon	Airspace of defined dimensions identified by an area on the surface of the earth wherein activities are confined
Runway	36	Line	Rectangular area on a land airport prepared for the landing and takeoff runs of aircraft along its length
SID	37	Multipoint	Standard Instrument Departure - preplanned instrument flight rule (IFR) for air traffic control departure procedure
SpecialUse Airspace	38	Point	Airspace of defined dimensions wherein activities are confined because of their nature and/or wherein limitations may be imposed upon aircraft operations that are not a part of those activities.
STAR	39	Multipoint	Standard Terminal Arrival - preplanned instrument flight rule (IFR) air traffic control arrival procedure
SupplTerminalData	40	Point	Supplemental terminal data
TerminalProcClimb	41	Point	Terminal Procedure Climb - Min or ATC Climb rates
TerminalProcFeedRoute	42	LineString	Terminal Procedure Feeder Route - A route depicted on Instrument Approach Procedures to designate routes for

			aircraft to proceed from the en route structure to the Initial Approach Fix
TerminalProcMin	43	Point	Terminal Procedure Minima - Height minima data for Terminal Procedure
VFRRoute	44	LineString	Preplanned arrival or departure routes for helicopters or light fixed wing aircraft to specified airports or heliports using/in Visual Flight Rules (VFR
VFRRouteSegment	45	LineString	Segment of a VFR Route
Waypoint	46	Point	Predetermined geographical position, used for route or instrument approach definition or progress reporting purposes

5.7.2.1 Default Read Value

Simulator client-devices should assume an empty tile when data is not available.

5.7.2.2 Default Write Value

The files associated with this dataset for the area covered by the geocell need not be created if the source data is not available.

5.7.3 Tiled GSFeature Dataset

A GSFeature is a geospecific (point, linear, or polygon) feature whose optional modeled representation is also geospecific.

The GSFeature Dataset provides the position, size, orientation (points), shape (linear and polygons), type and attribution of point, lineal, and polygon features. It is subdivided into the following components:

- CS1 = 001: Man-made features
- CS1 = 002: Natural features
- CS1 = 003: Trees features
- CS1 = 004: Airport features
- CS1 = 005: Environmental lights

Table 5-29: Component Selectors for GSFeature Dataset lists all of the components of the dataset. The allocation of CDB attributes to each of the components is prescribed by Table 5-27: Allocation of CDB Attributes to Vector Datasets.

It is customary in many simulation applications to represent street lighting as points of lights; as a result, Airport and Environmental light points can be entirely described by their feature position and attribution information and thus, do not have additional “modeled” data.

The modeling of geospecific trees is permitted when required to represent a specific geographic area; however, it is understood that the majority of the times, geotypical trees will be sufficient.

Table 5-29: Component Selectors for GSFeature Dataset

CS1	CS2	Component Name
001	001	Man-made point features
	002	Man-made point features class-level attributes
	016	Man-made point features extended-level attributes
	003	Man-made lineal features
	004	Man-made lineal features class-level attributes
	017	Man-made lineal features extended-level attributes
	005	Man-made polygon features
	006	Man-made polygon features class-level attributes
	018	Man-made polygon features extended-level attributes
002	001	Natural point features
	002	Natural point features class-level attributes
	016	Natural point features extended-level attributes
	003	Natural lineal features
	004	Natural lineal features class-level attributes
	017	Natural lineal features extended-level attributes
	005	Natural polygon features
	006	Natural polygon features class-level attributes
	018	Natural polygon features extended-level attributes
003	001	Trees point features
	002	Trees point features class-level attributes
	016	Trees point features extended-level attributes
004	001	Airport light point features
	002	Airport light point features class-level attributes
	016	Airport light point features extended-level attributes
	003	Airport lineal features
	004	Airport lineal features class-level attributes
	017	Airport lineal features extended-level attributes
	005	Airport polygon features
	006	Airport polygon features class-level attributes
	018	Airport polygon features extended-level attributes
005	001	Environmental light point features
	002	Environmental light point features class-level attributes
	016	Environmental light point extended-level attributes

5.7.3.1 Default Read Value

Simulator client-devices should assume an empty tile when data is not available.

5.7.3.2 Default Write Value

The files associated with this dataset for area covered by tile at a given LOD need not be created if the source data is not available.

5.7.4 Tiled GTFeature Dataset

A GTFeature is a geotypical (point, lineal, or polygon) feature whose optional modeled representation is also geotypical.

The GTFeature Dataset provides the position, size, orientation (points), shape (lineal and polygons), type and attribution of point, lineal, and polygon features. It is subdivided into the following components:

- CS1 = 001: Man-made features
- CS1 = 002: Trees features
- CS1 = 003: Moving model location features

Table 5-30: Component Selectors for GTFeature Dataset lists all of the components of the dataset. The allocation of CDB attributes to each of the components is prescribed by Table 5-27: Allocation of CDB Attributes to Vector Datasets.

The Moving model location features component is used to permanently position moving models (e.g., position a row of parked tanks or aircrafts on a runway). When referenced and positioned in this manner, these models cannot be moved and articulated during the simulation.

Table 5-30: Component Selectors for GTFeature Dataset

CS1	CS2	Component Name
001	001	Man-made point features
	002	Man-made point features class-level attributes
	016	Man-made point features extended-level attributes
	003	Man-made lineal features
	004	Man-made lineal features class-level attributes
	017	Man-made lineal features extended-level attributes
	005	Man-made polygon features
	006	Man-made polygon features class-level attributes
	018	Man-made polygon features extended-level attributes
002	001	Tree point features
	002	Tree point features class-level attributes
	016	Tree point features extended-level attributes
	003	Tree lineal features
	004	Tree lineal features class-level attributes
	017	Tree lineal features extended-level attributes
	005	Tree polygon features
	006	Tree polygon features class-level attributes
	018	Tree polygon features extended-level attributes
003	001	Moving Model location features
	002	Moving Model location features class-level attributes
	016	Moving Model location features extended-level attributes

5.7.4.1 Default Read Value

Simulator client-devices should assume an empty tile when data is not available.

5.7.4.2 Default Write Value

The files associated with this dataset for area covered by tile at a given LOD need not be created if the source data is not available.

5.7.5 Tiled GeoPolitical Feature Dataset

The GeoPolitical Feature dataset is used to provide information on the location, size and shape of abstract locations, lines and areas with respect to the surface of the earth. Generally speaking, the objects found in this dataset have no real-world physical representation (e.g., no physical characteristics such as mass) and correspond to abstract concepts (such as airspace, country boundary, danger zone).

The GeoPolitical Feature dataset is subdivided into the following components.

Table 5-31: Component Selectors for GeoPolitical Feature Dataset

CS1	CS2	Component Name
001	001	Boundary point features
	002	Boundary point features class-level attribute
	016	Boundary point features extended-level attribute
	003	Boundary lineal features
	004	Boundary lineal features class-level attribute
	017	Boundary lineal features extended-level attribute
	005	Boundary polygon features
	006	Boundary polygon features class-level attribute
	018	Boundary polygon features extended-level attribute
002	001	Location point features
	002	Location point features class-level attribute
	016	Location point features extended-level attribute
	003	Location lineal features
	004	Location lineal features class-level attribute
	017	Location lineal features extended-level attribute
	005	Location polygon features
	006	Location polygon features class-level attribute
	018	Location polygon features extended-level attribute
003	001	Constraint point features
	002	Constraint point features class-level attribute
	016	Constraint point features extended-level attribute
	003	Constraint lineal features
	004	Constraint lineal features class-level attribute
	017	Constraint lineal features extended-level attribute
	005	Constraint polygon features
	006	Constraint polygon features class-level attribute

5.7.5.1 Boundary and Location Features

GeoPolitical Polygon features are essentially used for closed surface related attributions whereas GeoPolitical Lineal features are used to model open areas as boundary delineations. When coarse specific locations are stored such as countries or cities, GeoPolitical Point features are used to locate the approximate geometric center of the related feature.

Some less-commonly used features could be 1) GeoPolitical Boundaries stored as Point features, or 2) GeoPolitical Locations stored as Lineals or Polygons. The first case could be used to represent a simple boundary consisting of a single radius as given by the Point feature. The second case could be used to represent a city location with detailed Polygon or Lineal vertices.

Figure 5-33: Example of International Boundaries, Figure 5-34: Example of City Locations, and Figure 5-35: Example of Major Cities and Time Zone Boundaries, all depict some sample cases where the GeoPolitical Feature dataset components can be used for.



Figure 5-33: Example of International Boundaries



Figure 5-34: Example of City Locations

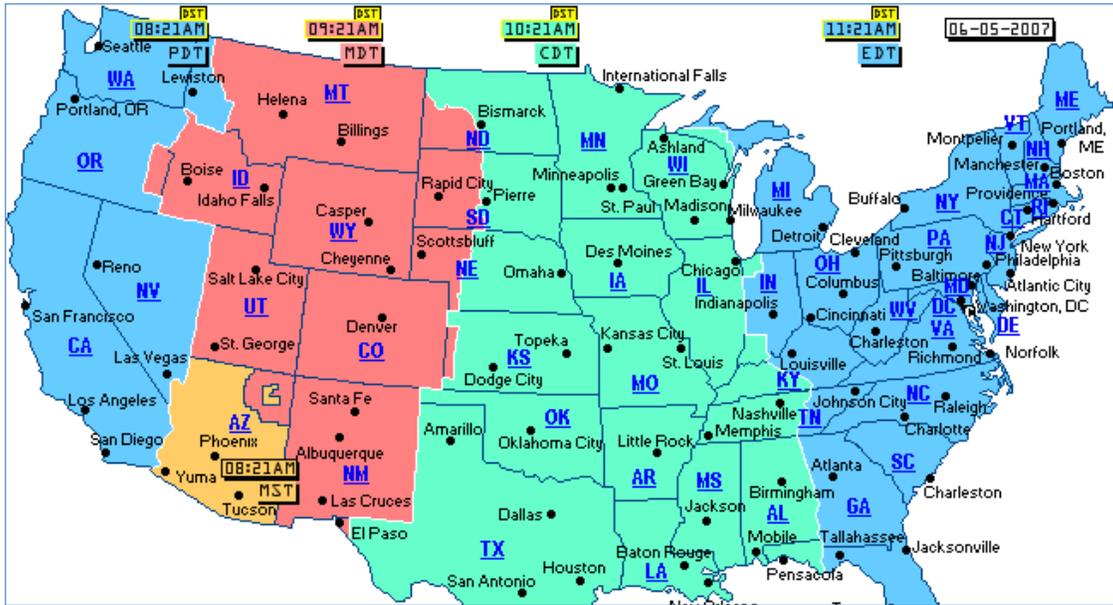


Figure 5-35: Example of Major City Locations and Time Zone Boundaries

5.7.5.2 Elevation Constraint Features

There are many instances where modelers may wish to take advantage of the availability of position and altitude of cultural features in order to locally control the terrain elevation data at a point, along a specified contour line or within a given area. This operation is usually performed off-line by the modeler and requires that the Elevation dataset be edited and re-generated offline.

Requirements Class - Elevation Constraints (120-123)	
/req/core/tiled-elevation-constraints	
Target type	Operations
Dependency	Various XML schema
Requirement 121	/req/core/elevation-constraint-ahgt
Requirement 122	/req/core/hypsography-elevation-offline
Requirement 123	/req/core/hypsograph-vector-types

In addition to this approach, the CDB standard provides a Constraint Features component to the GeoPolitical Feature dataset, whose vector data is designed to instruct client-devices to *runtime-constrain* the Elevation dataset to a set of prescribed elevation values (thereby obviating the need to offline re-generate the Elevation dataset).

At runtime, client-devices should apply the elevation constraints to the selected Terrain Elevation component of the Elevation dataset.

The Constraint Features component provides modelers the ability to accurately control terrain elevation profiles even if the Elevation dataset consists in a uniform grid of modest resolution. Each of these features is associated with vertices which define elevation at the supplied geographic coordinates. This approach approximates Terrain Irregular Networks (TINs). The Constraint Features have the following Feature Attribute Codes (feature codes):

- **Elevation Constraint Point (CA099-000):** In the case of PointZ and MultiPointZ features, the points are used by the client-device to control the terrain elevation.

Requirement 121

<http://www.opengis.net/spec/cdb/1.0/core/elevation-constraint-ahgt>

The Feature's AHGT attribute *SHALL* be set to TRUE. Vector data features implemented as Point, PointM, MultiPoint, and MultiPointM are ignored.

Elevation Constraint Line (CA099-001): In the case of PolyLineZ features, the lines *SHALL* be used by the client-device to control the terrain elevation. The Feature's AHGT attribute *SHALL* be set to TRUE. Vector features implemented as PolyLine and PolyLineM are ignored.

Elevation Constraint Area (CA099-002): In the case of PolygonZ and MultiPatch features, the areas *SHALL* be used by the client-device to control the terrain elevation. The Feature's AHGT attribute *SHALL* be set to TRUE. Vector features implemented as Polygon and PolygonM are ignored.

The Data Dictionary of the CDB standard also makes provision for the representation of many hypsography features within the Geopolitical Dataset (e.g., contour lines, ridge lines, valley lines, spot elevation). By virtue of their semantics, these features have no associated modeled representation. The modeler can use these hypsography features to control the generation of the Terrain Elevation grid during the off-line CDB compilation process. This terrain constraining operation can be performed as the CDB is "assembled and compiled" by the SE tools. Note that runtime client-devices are not required to constrain the Terrain Elevation Dataset to hypsography features.

Requirement 122

<http://www.opengis.net/spec/cdb/1.0/core/hypsography-elevation-offline>

When performed off-line, hypsography features *SHALL* have AHGT set to True, thereby instructing the SE Tools to constrain the terrain elevation using the supplied (latitude, longitude, and elevation) coordinates.

Requirement 123

<http://www.opengis.net/spec/cdb/1.0/core/hypsograph-vector-types>

The allowed feature types *SHALL* be of type PointZ, MultiPointZ, PolyLineZ, PolygonZ, or MultiPatch.

While hypsography features can be used by the off-line tools to control the terrain skinning process, these features can be instead converted into Constraint Features, thereby deferring the terrain constraining process to runtime client-devices. This provides modelers the ability to accurately control terrain elevation profiles even if the Elevation dataset consists in a uniform grid of modest resolution. In effect, the Constraint Features provides a storage-efficient means of capturing terrain contours without having to revert to high resolution terrain grids.

The application of constraint features to uniform and non-uniform gridded terrain elevation dataset is discussed in Volume 7: CDB Data Model Guidance (formerly Appendix A). See sections 6.6 and 6.7.

5.7.5.3 Default Read Value

Simulator client-devices should assume an empty tile when data is not available.

5.7.5.4 Default Write Value

The files associated with this dataset for area covered by tile at a given LOD need not be created if the source data is not available.

5.7.6 Tiled RoadNetwork Dataset

Requirements Class - Tiled Road Networks (124)	
/req/core/tiled-road-network	
Target type	Operations
Dependency	Various XML schema
Requirement 124	/req/core/tiled-roadnetwork

Requirement 124
http://www.opengis.net/spec/cdb/1.0/core/tiled-roadnetwork
The RoadNetwork Dataset <i>SHALL</i> be used to specify all of the road ⁴⁶ networks.

The points that make up the RoadNetwork lineals are primarily used to establish the road network topology. However, additional points can be used to more accurately define the actual path of the RoadNetwork lineals between each of the junction points of the network; alternately, points can be used to precisely specify the position of features along the RoadNetwork lineal. The altitude of each point is optional and specifies the ground level when present.

It is possible to associate an explicitly modeled representation (via the MODL and MODT attributes) to each RoadNetwork lineal. When provided, client-devices should instantiate the modeled representation for each point along the RoadNetwork lineal. Alternately, it is possible to specify a distinct modeled representation for each point along the RoadNetwork lineal feature by assigning a distinct RoadNetwork Figure Point to each point of the RoadNetwork lineal feature.

Table 5-32: Component Selectors for RoadNetwork Dataset lists all of the RoadNetwork Dataset components. The allocation of CDB attributes to each of the RoadNetwork dataset components is prescribed by Table 5-27: Allocation of CDB Attributes to Vector Datasets.

Table 5-32: Component Selectors for RoadNetwork Dataset

CS1	CS2	Component Name
-----	-----	----------------

⁴⁶ Within the context of the CDB, the term road encompasses a broad gamut of networks implemented as long, narrow stretch of smoothed or paved surfaces, made for traveling by foot, motor vehicle, carriage, etc., between two or more points. Examples of road networks are highways, interstates, roads, boulevards, streets, etc. It excludes railways.

001	011	Road/Airport Network Tile Connection 2D relationship
	015	Road/Airport Network Dataset Connection 2D relationship
002	003	Road Network lineal features
	004	Road Network lineal features class-level attributes
	017	Road Network lineal features extended-level attributes
	007	Road Network lineal figure point features
	008	Road Network lineal figure point class-level attributes
	019	Road Network lineal figure point extended-level attributes
003	003	Airport Network lineal features
	004	Airport Network lineal features class-level attributes
	017	Airport Network lineal features extended-level attributes
	005	Airport Network polygon features
	006	Airport Network polygon features class-level attributes
	018	Airport Network polygon features extended-level attributes

5.7.6.1 Default Read Value

Simulator client-devices should assume an empty tile when data is not available.

5.7.6.2 Default Write Value

The files associated with this dataset for area covered by tile at a given LOD need not be created if the source data is not available.

5.7.7 Tiled RailRoadNetwork Dataset

Requirements Class - Tiled Railroad Networks (125)	
/req/core/tiled-railroad-network	
Target type	Operations
Dependency	Various XML schema
Requirement 125	/req/core/tiled-railroadnetwork

Requirement 125
http://www.opengis.net/spec/cdb/1.0/core/tiled-railroadnetwork

The RailRoadNetwork Dataset *SHALL* be used to specify all of the railroad⁴⁷ networks.

The points that make up the RailRoadNetwork lineals are primarily used to establish the railroad network topology. However, additional points can be used to more accurately define the actual path of the RailRoadNetwork lineals between each of the junction points of the network; alternately, points can be used to precisely specify the position of features along the RailRoadNetwork lineal. The altitude of each point is optional and specifies the ground level when present.

It is possible to associate an explicitly modeled representation (via the MODL and MODT attributes) to each RailRoadNetwork lineal. When provided, client-devices should instance the modeled representation for each point along the RailRoadNetwork lineal. Alternately, it is possible to specify a distinct modeled representation for each point along the RailRoadNetwork lineal feature by assigning a distinct a RailRoadNetwork Figure Point to each point of the RailRoadNetwork lineal feature.

Table 5-33: Component Selectors for RailRoadNetwork Dataset lists RailRoadNetwork Dataset components. The allocation of CDB attributes to each of the RailRoadNetwork dataset components is prescribed by Table 5-27: Allocation of CDB Attributes to Vector Datasets.

Table 5-33: Component Selectors for RailRoadNetwork Dataset

CS1	CS2	Component Name
001	011	RailRoad Network Tile Connection 2D relationship point
	015	RailRoad Network Dataset Connection 2D relationship point
002	003	RailRoad Network lineal features
	004	RailRoad Network lineal features class-level attribute
	017	RailRoad Network lineal features extended-level attribute
	007	RailRoad Network lineal figure point features
	008	RailRoad Network lineal figure point class-level attribute
	019	RailRoad Network lineal figure point extended-level attribute

5.7.7.1 Default Read Value

Simulator client-devices should assume an empty tile when data is not available.

⁴⁷ Within the context of the CDB, the term railroads encompasses a broad gamut of networks implemented as roads that are composed of parallel steel rails supported by ties and providing a track for locomotive-drawn trains or other wheeled vehicles.

5.7.7.2 Default Write Value

The files associated with this dataset for area covered by tile at a given LOD need not be created if the source data is not available.

5.7.8 Tiled PowerLineNetwork Dataset

Requirements Class - Tiled Power Line Networks (126)	
/req/core/tiled-powerline-network	
Target type	Operations
Dependency	Various XML schema
Requirement 126	/req/core/tiled-powerlinenetwork

Requirement 126
http://www.opengis.net/spec/cdb/1.0/core/tiled-powerlinenetwork
The PowerLineNetwork Dataset <i>SHALL</i> be used to specify powerline ⁴⁸ networks.

The points that make up the PowerLineNetwork lineals are primarily used to establish the network topology. However, additional points can be used to more accurately define the actual path of the PowerLineNetwork lineals between each of the junction points of the network; alternately, points can be used to precisely specify the position of the poles, pylons, etc. of the PowerLineNetwork lineal. The altitude of each point is optional and specifies the ground level when present.

It is possible to associate an explicitly modeled representation (via the MODL and MODT attribute) to each PowerLineNetwork lineal. When provided, client-devices should instance the modeled representation for each point along the PowerLineNetwork lineal. Alternately, it is possible to specify a distinct modeled representation for each point along the PowerLineNetwork lineal feature by assigning a distinct a PowerLineNetwork Figure Point to each point of the PowerLineNetwork lineal feature. Allowed model representations are as per Section 5.7.1.4, Explicitly Modeled Representations. In that case of wired- or cabled-networks, the model must

⁴⁸ Within the context of the CDB, the term powerline encompasses a broad gamut of networks implemented through the use of wires, cables and/or pipes.

include the wire Attach Points⁴⁹. Note also that client-devices are also required to automatically orient each instance of the modeled representation along the path of the PowerLineNetwork lineal. In the case where the entrance and exit angles are not identical, the orientation should be the average of both.

Table 5-34: Component Selectors for PowerLineNetwork Dataset, lists all of the PowerLineNetwork Dataset components. The allocation of CDB attributes to each of the PowerLineNetwork dataset components is prescribed by Table 5-27: Allocation of CDB Attributes to Vector Datasets.

Table 5-34: Component Selectors for PowerLineNetwork Dataset

CS1	CS2	Component Name
001	011	PowerLineNetwork tile connection 2D relationship point
	015	PowerLineNetwork dataset connection 2D relationship point
002	003	PowerLineNetwork lineal features
	004	PowerLineNetwork lineal features class-level attribute
	017	PowerLineNetwork lineal features extended-level attribute
	007	PowerLineNetwork lineal figure point features
	008	PowerLineNetwork lineal figure point class-level attribute
	019	PowerLineNetwork lineal figure point extended-level attribute

5.7.8.1 Default Read Value

Simulator client-devices should assume an empty tile when data is not available.

5.7.8.2 Default Write Value

The files associated with this dataset for area covered by tile at a given LOD need not be created if the source data is not available.

5.7.9 Tiled HydrographyNetwork Dataset

Requirements Class - Tiled Hydrography Networks (127)	
/req/core/tiled-hydro-network	
Target type	Operations

⁴⁹Note that wires are not explicitly modeled in the CDB. As a result, client-devices are required to automatically model them when rendering PowerLineNetwork lineal features. Such modeling can be achieved using a principle known as the catenary, which is the shape of a perfectly flexible chain suspended by its ends and which is characterized solely by the gravity action.

Dependency	Various XML schema
Requirement 127	/req/ core /tiled-hydroneetwork

Requirement 127
http://www.opengis.net/spec/cdb/1.0/core/tiled-hydroneetwork
The HydrographyNetwork Dataset <i>SHALL</i> be used to specify hydrography ⁵⁰ networks.

The points that make up the HydrographyNetwork lineals are primarily used to establish the network topology. However, additional points can be used to more accurately define the actual path of the HydrographyNetwork lineals between each of the junction points of the network; alternately, points can be used to precisely specify the position of models of the HydrographyNetwork lineal. The altitude of each point is optional and specifies the ground level when present.

It is possible to associate an explicitly modeled representation (via the MODL and MODT attributes) to each HydrographyNetwork lineal. When provided, client-devices should instance the modeled representation for each point along the HydrographyNetwork lineal. Alternately, it is possible to specify a distinct modeled representation for each point along the HydrographyNetwork lineal feature by assigning a distinct a HydrographyNetwork Figure Point to each point of the HydrographyNetwork lineal feature. Allowed model representations are as per Section 5.3.1.4, Explicitly Modeled Representations.

Table 5-35: Component Selectors for HydrographyNetwork Dataset lists all of the components of the HydrographyNetwork dataset. The allocation of CDB attributes to each of the HydrographyNetwork dataset components is prescribed by Table 5-27: Allocation of CDB Attributes to Vector Datasets.

Table 5-35: Component Selectors for HydrographyNetwork Dataset

CS1	CS2	Component Name
001	011	HydrographyNetwork tile connection 2D relationship point
	015	HydrographyNetwork dataset connection 2D relationship point
002	003	HydrographyNetwork lineal features
	004	HydrographyNetwork lineal features class-level attribute

⁵⁰ Within the context of the CDB, the term hydrography encompasses a broad gamut of natural and man-made bodies of water such as oceans, lakes, rivers, canals, etc.

017	HydrographyNetwork lineal features extended-level attribute
005	HydrographyNetwork polygon features
006	HydrographyNetwork polygon features class-level attribute
018	HydrographyNetwork polygon features extended-level attribute
007	HydrographyNetwork lineal figure point features
008	HydrographyNetwork lineal figure point class-level attribute
019	HydrographyNetwork lineal figure point extended-level attribute
009	HydrographyNetwork polygon figure point features
010	HydrographyNetwork polygon figure point class-level attribute
020	HydrographyNetwork polygon figure point extended-level attribute

5.7.9.1 Default Read Value

Simulator client-devices should assume an empty tile when data is not available.

5.7.9.2 Default Write Value

The files associated with this dataset for area covered by tile at a given LOD need not be created if the source data is not available.

5.7.10 Tiled Vector Composite Material Table (VCMT)

Requirements Class - Tiled Composite Materials Table (128)	
/req/core/tiled-vcmt	
Target type	Operations
Dependency	Various XML schema
Requirement 128	/req/core/vcmt

Requirement 128
http://www.opengis.net/spec/cdb/1.0/core/vcmt
The Vector Composite Material Table (VCMT) provides a list of the Composite Materials shared by all vector datasets. There <i>SHALL</i> be one VCMT for each CDB Geocell.

Table 5-36: Vector Composite Material Table Component

CS1	CS2	File Extension	Dataset Component Name	Dataset Component Description
Dataset 200, VectorMaterial				
001	001	*.xml	Vector Composite Material Table (VCMT)	<p>The VCMT can be referenced by the CMIX attribute of the following datasets:</p> <ul style="list-style-type: none"> • 100_GSFeature • 101_GTFeature • 102_GeoPolitical • 201_RoadNetwork • 202_RailRoadNetwork • 203_PowerLineNetwork • 204_HydrographyNetwork

5.7.10.1 Data Type

The Vector Composite Material Table follows the XML syntax described in Section 2.5.2.2, Composite Material Tables (CMT).

5.7.10.2 Default Read Value

The default values for the Vector Composite Material Table (Default_Material_Layer) can be found in \CDB\Metadata\Defaults.xml and can be provided to the client-devices on demand. If the default information cannot be found within the \CDB\Metadata\Defaults.xml file, the CDB Specification recommends defaulting to single substrate composite material whose base material is Default_Base_Material (BM_LAND-MOOR).

5.7.10.3 Default Write Value

The files associated with the Vector Composite Material Table for the area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.

5.8 Tiled Model Datasets

5.8.1 Tiled GSModel Datasets

Table 5-37, Component Selectors for GSModel Datasets, provides the dataset codes, the file extension types and associated component selector values of GSModels geometry, GSModelInteriors geometry and their associated Composite Material Tables (CMT).

Table 5-37: Component Selectors for GSModel Datasets

CS1	CS2	File Extension	Component Name	Component Description
Dataset 300, GSModelGeometry Dataset 305, GSModelInteriorGeometry				
001	001..999	*.flt	Individual Geometry	One model containing the geometry of the shell or multiple files to represent the interior of a GSModel as described in Chapter 6.
001	001	*.zip	Geometry Archive	One archive regrouping individual model geometry model files of a Tile-LOD.
Dataset 303, GSModelDescriptor Dataset 307, GSModelInteriorDescriptor				
001	001	.xml	Individual Descriptor	Provides the metadata associated with individual GSModels. See section 6.14, Metadata, for a description of the content. <u>NOTE:</u> A model descriptor includes a Composite Material Table for the exclusive use by its corresponding model geometry datasets above. This CMT is not to be confused with the GSModelCMT dataset below.
001	001	.zip	Descriptor Archive	An archive of all model descriptors of a Tile-LOD.
Dataset 301, GSModelTexture Dataset 306, GSModelInteriorTexture				
-	-	*.rgb	Individual Texture	Individual base and subordinate textures applied on individual or tiled models of a Tile-LOD, see the complete list in section 5.3, CDB Model Textures.
001	001	*.zip	Texture Archive	An archive of all base and subordinate textures of a Tile-LOD.
Dataset 304, GSModelMaterial Dataset 308, GSModelInteriorMaterial				
001	001..255	*.tif	Composite Material Index	Each texel is an index into the GSModelCMT. Component selector 2 is the layer number.
002	001..254	*.tif	Composite Material Mixture	Each texel indicates the proportion (between 0.0 and 1.0) of the composite material found in the corresponding material layer. Component Selector 2 is the layer number. When the texels are of integral types, they are scaled to the range 0.0 to 1.0.
001	001	*.zip	Composite Material Archive	An archive of all layers of indices and mixtures of a Tile-LOD.
Dataset 309, GSModelCMT				
001	001	*.xml	Composite Material Table	Provides a local Composite Material Table per geocell; it contains the definition of the composite materials referenced by the model material datasets above. Its format is as specified in section 2.5.2.2,

				Composite Material Tables (CMT)
--	--	--	--	---------------------------------

5.8.1.1 GSModel Archive Size Limit

The size of any GSModel archive is limited to 32 MB to permit reading, processing and writing the file at runtime.

5.8.2 Tiled T2DModel Datasets

Table 5-38 provides the component selectors for the T2DModel datasets.

Table 5-38: Component Selectors for T2DModel Datasets

CS1	CS2	File Extension	Component Name	Component Description
Dataset 310, T2DModelGeometry				
001	001	*.flt	Model Geometry	One model file containing the geometry of the T2DModel of a Tile-LOD. The content of the file is explained in Volume 6 – CDB Rules for Encoding Data.

Annex A

(normative)

Abstract Conformance Test Suite

A.1 Conformance Test Class: OGC CDB Core Standard (Core)

This section describes conformance test for the OGC CDB Standard Core. These abstract test cases describe the conformance criteria for verifying the structure and content of any data store or database claiming conformance to the CDB standard.

The conformance class id is “<http://www.opengis.net/spec/CDB/1.0/conf/>” and all of the other conformance tests URLs are created in this path. Another issue that the reader should pay attention to is the test method. When the test method is assigned with “Visual”, it means that the purpose of the test should be “visually” investigate the file contents, image, or other content.

A.1.1 General CDB Data Store and Implementation

The following conformance class is designed to determine if a CDB implementation include all elements of the CDB Core Data Model as defined in Annex A.

Conformance Class	<i>/conf/core/model</i>	
Requirements	<i>/req/core/model</i>	
Dependency	Target system operating and file management system, Annex A	
Test 1	Test purpose	Verify that all elements of the CDB Core Data Model are included in the implementation as defined in Annex A.
	Test method	Pass if all of the test cases in Annex A conformance test suite class are passed.
	Test type	Conformance

A.1.2 Platform

The following conformance class is designed to determine if the target hardware system (laptop, desktop, etc) has the resources necessary to store, manage, update, and access a CDB data store.

Conformance Class	<i>/conf/core/platform</i>
--------------------------	----------------------------

Requirements	/req/core/platform	
Dependency	Operating System	
Dependency	Hardware	
Dependency	File Management system	
Test 5	/conf/platform/literal-case	
	Requirement	req/core/literal-case
	Test purpose	Verify the CDB implementation including filenames and directory paths supports the literal case rules and semantic meaning as specified in the file naming conventions.
	Test method	Visual
	Test type	Conformance

A.1.3 General Data Representation

The following conformance class is designed to determine if the general data representation requirements are met.

Conformance Class	/conf/core/data-representation	
Requirements	/req/core/data-representation	
Dependency	General data representation unit, coordinate reference system and JPEG 2000 compression algorithm	
Test 6	/conf/core/data-representation/raster-imagery-compression	
	Requirement	/req/core/raster-imagery-compression
	Test purpose	Verify that the raster imagery datasets are stored and compressed as JPEG 2000 specification (Annex C Volume 2).
	Test method	Visual
	Test type	Conformance
Test 7	/conf/core/data-representation/uom	
	Requirement	/req/core/uom
	Test purpose	Verify that the all units of measure in the data store are in meters.
	Test method	Visual

	Test type	Conformance
Test 8	/conf/core/data-representation/crs	
	Requirement	/req/core/crs
	Test purpose	Verify the geographic coordinates in CDB are expressed using WGS-84 (World Geodetic System 1984). If there is an altitude associated with the coordinates, test that the altitude is relative to the reference ellipsoid.
	Test method	Visual
	Test type	Conformance
Test 9	/conf/core/data-representation/crs-client	
	Requirement	req/core/crs-client
	Test purpose	Verify that each implementation of the simulator client-devices accessing the CDB geospatial data uses the WGS-84 geodetic coordinate system.
	Test method	Visual
	Test type	Conformance
Test 10	/conf/core/data-representation/coordinates	
	Requirement	req/core/coordinates
	Test purpose	Verify that Coordinates are described using the decimal degree format bounded by $\pm 90^\circ$ latitude, and $\pm 180^\circ$ longitude respectively
	Test method	Visual
	Test type	Conformance

A.1.4 Structure-Tiling Model

The following conformance class is designed to determine if the CDB structure model has the necessary requirements.

Conformance Class	/conf/core/cdb-structure-tiles-lod
Requirements	/req/core/cdb-structure-tiles-lod

Dependency	CDB data store structure and model and coordinate reference system	
Test 11	/conf/core/cdb-structure-tiles-lod/geocell-extent	
	Requirement	/req/core/geocell-extent
	Test purpose	Verify the CDB Geocell extent is a whole degree along with the proper longitudinal axis
	Test method	Visual
	Test type	Conformance
Test 12	/conf/core/cdb-structure-tiles-lod/geocell-length	
	Requirement	/req/core/geocell-length
	Test purpose	Verify the CDB Geocell length is a whole factor of 180
	Test method	Visual
	Test type	Conformance
Test 13	/conf/core/cdb-structure-tiles-lod/tile-sizes	
	Requirement	/req/core/tile-sizes
	Test purpose	Verify the tile sizes is 1024 × 1024 grid size for all of the positive LODs
	Test method	Visual
	Test type	Conformance
Test 14	/conf/core/cdb-structure-tiles-lod/lod-area-coverage	
	Requirement	req/core/lod-area-coverage
	Test purpose	Verify a tile from LOD –10 to LOD 0 occupy the area of exactly one CDB Geocell and a tile from LOD 1 and up recursively is subdivided into smaller tiles and covers smaller area based on the table 2-4
	Test method	Visual
	Test type	Conformance
Test 15	/conf/core/cdb-structure-tiles-lod/hierarchy	
	Requirement	req/core/hierarchy

	Test purpose	Verify the LOD hierarchy of all tiled datasets is complete for every CDB Geocell, although the number of Tile-LODs is different for each Geocell.
	Test method	Visual
	Test type	Conformance
Test 16	/conf/core/cdb-structure-tiles-lod/tile-lod-replacement	
	Requirement	req/core/tile-lod-replacement
	Test purpose	Verify that for negative LODs, a tile at LODn exactly replaces a tile at LODn-1, and for positive LODs, a tile at LODn is replaced by 4 tiles at LODn+1
	Test method	Visual
	Test type	Conformance
Test 17	/conf/core/cdb-structure-tiles-lod/lod-organization-resolution	
	Requirement	req/core/lod-organization-resolution
	Test purpose	Verify the geometry, texture, and signature datasets of 3D models is organized into levels of details (LOD) based on their resolutions
	Test method	Visual
	Test type	Conformance

A.1.5 Light Naming

The following conformance class is designed to determine if any modeled light point is described based on the comprehensive set of light hierarchies suited to the needs of visual simulation.

Conformance Class	/conf/core/light-name	
Requirements	/req/core/light-name	
Dependency		
Test 17	Test purpose	Verify that name of a light is based on the naming hierarchy style.

	Test method	Pass if the light nam is visually conformed with a naming hierarchy and starting from the top-most level of the hierarchy, concatenate each of the names encountered with the backslash“\” character.
	Test type	Conformance

A.1.6 Light Name Hierarchy

The following conformance class is designed to determine if a light type is added to the hierarchy of light names has the necessary requirements.

Conformance Class	/conf/core/light-name-hierarchy	
Requirements	/req/core/light-name-hierarchy	
Dependency	XML	
Dependency	Light Name	
Dependency	XML Schema – Part 2	
Test 18	req/core/light-name-hierarchy/type name	
	Requirement	/req/core/light-name-hierarchy/type name
	Test purpose	Verify that the modeler or the simulator vendor creates a new light type name and light code which follow an XML schema specified in Req. 18.
	Test method	Inspect whether the light code is conformant to light.XML schema located in the CDB schema folder.
	Test type	Conformance
Test 19	/conf/core/cdb-structure-tiles-lod/geocell-length	
	Requirement	/req/core/light-name-hierarchy/type-name-definition
	Test purpose	Verify that the modeler or the simulator vendor provides an appropriate definition for the light type name.
	Test method	Inspect whether the light type name is conformant to light.XML schema located in the CDB schema folder.
	Test type	Conformance
Test 20	/conf/core/light-name-hierarchy/insert	
	Requirement	req/core/light-name-hierarchy/insert
	Test purpose	Verify that the modeler or the simulator vendor inserts the new light type into the light name hierarch
	Test method	Inspect whether the light xml schema file created/edited by the

		simulator/vendor is conformant to light.XML schema located in the CDB schema folder.
	Test type	Conformance
Test 21	/conf/core/light-name-hierarchy/edit	
	Requirement	req/core/light-name-hierarchy/edit
	Test purpose	Verify that the modeler or the simulator vendor edits the Lights.xml metadata file to reflect the change to the light name hierarchy.
	Test method	Inspect whether the light xml schema file edited by the simulator/vendor is conformant to light.XML schema located in the CDB schema folder.
	Test type	Conformance

A.1.7 Materials

The following conformance class is designed to deal with the handling of materials that make up the synthetic environment when adding a new model component.

Conformance Class	/conf/core/materials	
Requirements	/req/core/materials	
Dependency	Material	
Dependency	XML	
Dependency	CDB data store structure and model	
Test 22	/conf/core/materials/composite-materials-resolution	
	Requirement	/req/core/composite-materials-resolution
	Test purpose	Verify that references to composite materials resolves down to one or more of the stated CDB Base Materials.
	Test method	Inspect whether the composite materials are conformant to material.XML schema located in the CDB schema folder.
	Test type	Conformance
Test 23	/conf/core/materials/base-material-name	
	Requirement	/req/core/base-material-name
	Test purpose	Verify the base material's name begins with "BM_" followed by a unique arbitrary string based on specific conventions.

	Test method	Visual
	Test type	Conformance
Test 24	/conf/core/materials/primary-substrate	
	Requirement	/req/core/primary-substrate
	Test purpose	Verify that if there is a Primary_Substrate, there is only one.
	Test method	Visual
	Test type	Conformance
Test 25	/conf/core/materials/cdb-structure-tiles-lod/lod-area-coverage	
	Requirement	req/core/base-material-coverage
	Test purpose	Verify that the base materials of each substrate is listed in decreasing order of weighting.
	Test method	Visual
	Test type	Conformance
Test 26	/conf/core/materials/composite-material-tag	
	Requirement	req/core/composite-material-tag
	Test purpose	Verify that each composite material <i>is</i> tagged with a specific convention.
	Test method	Visual
	Test type	Conformance
Test 27	/conf/core/materials/sem-base-materials	
	Requirement	req/core/sem-base-materials
	Test purpose	Verify that any SEM has a corresponding Base Material for each of the CDB Base Materials.
	Test method	Visual
	Test type	Conformance

Test 28	/conf/core/materials/generation-materials-3d	
	Requirement	req/core/generation-materials-3d
	Test purpose	Verify the zones or selected polygons are tagged with the appropriate materials in 3D models.
	Test method	Visual
	Test type	Conformance

A.1.8 CDB Root Directory

The following conformance class is designed to deal with the handling of top-level directory structure of the CDB from the root directory.

Conformance Class	/conf/core/file-structure	
Requirements	/req/core/cdb-root-requirements	
Dependency	CDB data store structure and model	
Test 29	/conf/core/cdb-root-requirements/root-file-hierarchy	
	Requirement	req/core/root-file-hierarchy
	Test purpose	Verify the files stored within a CDB data store are under the root directory or within a subdirectory under the root directory
	Test method	Visual
	Test type	Conformance
Test 30	/conf/core/cdb-root-requirements/root-give-path	
	Requirement	req/core/root-give-path
	Test purpose	Verify that run-time applications are given the path and device on which the CDB is stored in order to access the CDB.
	Test method	Visual
	Test type	Conformance
Test 31	/conf/core/cdb-root-requirements/root-access-version	
	Requirement	req/core/root-access-version
	Test purpose	Verify that the CDB Standard also has provisions for the handling of multiple, incremental versioning of the CDB.

	Test method	Visual
	Test type	Conformance
Test 32	/conf/core/cdb-root-requirements/root-version-default	
	Requirement	req/core/root-version-default
	Test purpose	Verify that If no change is encountered in any of the incremental versions, the applications use the content of the active default CDB.
	Test method	Visual
	Test type	Conformance

A.1.9 Version Metadata File

The following conformance class is designed to determine if Version.xml has the necessary requirements.

Conformance Class	/conf/core/metadata-version	
Requirements	/req/core/metadata-version	
Dependency	XML	
Test 33	Test purpose	Verify that CDB Version has a metadata file available in a specific folder
	Test method	Visually inspect whether each CDB Version has a metadata file called "Version.xml" available in "\CDB\Metadata"
	Test type	Conformance

A.1.10 FLIR Metadata File

The following conformance class is designed to determine if Lights_FLIR.xml have the necessary requirements.

Conformance Class	/conf/core/metadata-flir	
Requirements	/req/core/metadata-flir	
Dependency	XML	

Test 34	Test purpose	Verify if “FLIR” client device has a client specific metadata file available in a specific folder
	Test method	Visually inspect whether if each “FLIR” client device has a client specific metadata file called “Lights_FLIR.xml” available in a specific folder “\CDB\Metadata\ “
	Test type	Conformance

A.1.11 Data Store Version Directory Structure

The following conformance class is designed to deal with the handling files and the directory structure of CDB Versions.

Conformance Class	/conf/core/versioning	
Requirements	/req/core/versioning	
Dependency	File structure	
Test 35	/conf/core/versioning/cdb-not-in-root-directory	
	Requirement	/req/core/cdb-not-in-root-directory
	Test purpose	Verify that a CDB Version is stored directly in the root directory of a disk device or volume
	Test method	Visual
	Test type	Conformance
Test 36	/conf/core/versioning/cdb-version-path	
	Requirement	/req/core/cdb-version-path
	Test purpose	Verify that a CDB Version path name is acceptable while using within another CDB Version
	Test method	Visual
	Test type	Conformance
Test 37	/conf/core/versioning/cdb-version-entry-point-access	
	Requirement	/req/core/cdb-version-entry-point-access
	Test purpose	Verify that the path to boot CDB is provided to all client devices and run-time applications have access, directly or indirectly, to all disk devices and volumes as well as all paths to all linked CDB Versions simultaneously.
	Test method	Visual
	Test type	Conformance

Test 38	/conf/core/versioning/cdb-chain-max	
	Requirement	req/core/cdb-chain-max
	Test purpose	Verify that all CDB chains have no more than eight CDB versions
	Test method	Visual
	Test type	Conformance

A.1.12 Model Types

The following conformance class is designed to determine if cultural features of a CDB database have the necessary requirements.

Conformance Class	/conf/core/cultural-representation	
Requirements	/req/core/cultural-representation	
Dependency		
Test 39	Test purpose	Verify if the cultural features of a CDB data store are valid file type.
	Test method	Visually inspect weather the cultural features of a CDB data store are one of the following types of modeled representations: GTModel, GSModel, T2DModel & MModel.
	Test type	Conformance

A.1.13 Geospecific Model (GSModel) Storage

The following conformance class is designed to determine if GSModels have the necessary requirements.

Conformance Class	/conf/core/geospecific-storage	
Requirements	/req/core/geospecific-storage	
Dependency	Feature Code (FC)	
Test 40	Test purpose	Verify if geospecific models are stored in the \CDB\Tiles\ with a unique file name.
	Test method	Visually check if geospecific models are stored in the \CDB\Tiles\

		with a unique file name derived from the model's unique position, level-of-detail, and its feature code.
	Test type	Conformance

A.1.14b Geotypical Models LOD Resolution

Conformance Class	/conf/core/lod-organization-resolution	
Requirements	/req/core/lod-organization-resolution	
Dependency	NA	
Test 41	Test purpose	Verify if the geometry, texture, and signature datasets of 3D models is organized into levels of details (LOD) based on their resolutions.
	Test method	Visually check if the geometry, texture, and signature datasets of 3D models is organized into levels of details (LOD) based on their resolutions.
	Test type	Conformance

A.1.14 Geotypical Models (GTModel) Naming Conventions

The following conformance class is designed to determine if the GTModels have the necessary requirements.

Conformance Class	/conf/core/gtmodel-naming	
Requirements	/req/core/gtmodel-naming	
Dependency	Various XML schema	
Test 42	/conf/core/gtmodel-naming/texture-name	
	Requirement	/req/core/texture-name
	Test purpose	Verify that the name of 3D model textures is character string having a minimum of 2 characters and a maximum length of 32 characters and the first two characters are alphanumeric.
	Test method	Visual
	Test type	Conformance
Test 43	/conf/core/gtmodel-naming/texture-name-file-name	
	Requirement	/req/core/texture-name-file-name
	Test purpose	Verify that the acronym TNAM represents the texture name and is used to compose texture file and directory names and The following directory structure is used by CDB Model texture-related

		datasets: \\A\B\TNAM\
	Test method	Visual
	Test type	Conformance
Test 44	/conf/core/gtmodel-naming/lod-file-name	
	Requirement	/req/core/lod-file-name
	Test purpose	Verify that in the context of the CDB Standard, filenames and directory names are composed from the concept of LOD.
	Test method	Visual
	Test type	Conformance
Test 45	/conf/core/gtmodel-naming/gtmodel-directories	
	Requirement	/req/core/gtmodel-directories
	Test purpose	Verify that the “\CDB\GTModel” folder is the root directory of the GTModel library which is composed of the following datasets: GTModelGeometry,GTModelTexture,GTModelDescriptor,GTModelMaterial, GTModelCMT, GTModelInteriorGeometry, GTModelInteriorTexture,GTModelInteriorDescriptor,GTModelInteriorMaterial, GTModelInteriorCMT,GTModelSignature.
	Test method	Visual
	Test type	Conformance
Test 46	/conf/core/gtmodel-naming/gtmodelgeometry-entry-name	
	Requirement	/req/core/gtmodelgeometry-entry-name
	Test purpose	Verify that the files of the GTModelGeometry Entry File dataset is stored in level 4 of its directory structure and the names of the files adhere to the following naming convention: D500_Snnn_Tnnn_FeatureCode_FSC_MODL.fit.
	Test method	Visual
	Test type	Conformance
Test 47	/conf/core/gtmodel-naming/gtmodelgeometry-lod-name	
	Requirement	/req/core/gtmodelgeometry-lod-name

	Test purpose	Verify that the files of the GTModelGeometry Level of Detail dataset are stored in level 5 of its directory structure and the names of the files adhere to the following naming convention: D510_Snnn_Tnnn_LOD_FeatureCode_FSC_MODL.fit
	Test method	Visual
	Test type	Conformance
Test 48	/conf/core/gtmodel-naming/gtmodeldescriptor-name	
	Requirement	/req/core/gtmodeldescriptor-name
	Test purpose	Verify that the files of the GTModelDescriptor dataset are stored in level 4 of its directory structure and the names of the files adhere to the following naming convention: D503_Snnn_Tnnn_FeatureCode_FSC_MODL.xml
	Test method	Visual
	Test type	Conformance
Test 49	/conf/core/gtmodel-naming/gtmodeltexture-name	
	Requirement	/req/core/gtmodeltexture-name
	Test purpose	Verify that the names of the GTModelTexture files adhere to the following naming convention: D511_Snnn_Tnnn_LOD_TNAM.rgb
	Test method	Visual
	Test type	Conformance
Test 50	/conf/core/gtmodel-naming/gtmodelmaterial-name	
	Requirement	/req/core/gtmodelmaterial-name
	Test purpose	Verify that the names of the GTModelMaterial files adhere to the following naming convention: D504_Snnn_Tnnn_LOD_TNAM.tif
	Test method	Visual
	Test type	Conformance

Test 51	/conf/core/gtmodel-naming/gtmodelcmt-name	
	Requirement	/req/core/gtmodelcmt-name
	Test purpose	Verify that the names of the GTModelCMT files adhere to the following naming convention: D505_Snnn_Tnnn_TNAM.xml
	Test method	Visual
	Test type	Conformance
Test 52	/conf/core/gtmodel-naming/gtmodelinteriorgeometry-name	
	Requirement	/req/core/gtmodelinteriorgeometry-name
	Test purpose	Verify that the files of the GTModelInteriorGeometry dataset is stored in level 5 of its directory structure and the names of the files adhere to the following naming convention: D506_Snnn_Tnnn_LOD_FeatureCode_FSC_MODL.ftt
	Test method	Visual
	Test type	Conformance
Test 53	/conf/core/gtmodel-naming/gtmodelinteriordescriptor-name	
	Requirement	/req/core/gtmodelinteriordescriptor-name
	Test purpose	The files of the GTModelInteriorDescriptor dataset is stored in level 4 of the 5-level directory structure presented above. The names of the files adhere to the following naming convention: D508_Snnn_Tnnn_FC_FSC_MODL.xml
	Test method	Visual
	Test type	Conformance
Test 54	/conf/core/gtmodel-naming/gtmodelinteriortexture-name	
	Requirement	/req/core/gtmodelinteriortexture-name
	Test purpose	Verify that the names of the GTModelInteriorTexture files adhere to the following naming convention: D507_Snnn_Tnnn_LOD_TNAM.rgb

	Test method	Visual
	Test type	Conformance
Test 55	/conf/core/gtmodel-naming/gtmodelinteriormaterial-name	
	Requirement	/req/core/gtmodelinteriormaterial-name
	Test purpose	Verify that the names of the GTModelInteriorMaterial files adhere to the following naming convention: D509_Snnn_Tnnn_LOD_TNAM.tif
	Test method	Visual
	Test type	Conformance
Test 56	/conf/core/gtmodel-naming/gtmodelsignature-name	
	Requirement	/req/core/gtmodelsignature-name
	Test purpose	Verify that the names of the GTModelSignature files adhere to the following naming convention: D512_Snnn_Tnnn_LOD_FeatureCode_FSC_MODL.xxx
	Test method	Visual
	Test type	Conformance

A.1.15 Moving Model (MModel) Naming Conventions

The following conformance class is designed to determine if MModel library datasets have the necessary requirements.

Conformance Class	/conf/core/mmodel-naming	
Requirements	/req/core/mmodel-naming	
Dependency	Various XML schema	
Test 57	/conf/core/mmodel-naming/mmodel-root	
	Requirement	/req/core/mmodel-root
	Test purpose	Verify that the \CDB\MModel\ folder is the root directory of the

		MModel library and is composed of the following datasets. MModelGeometry, MModelDescriptor, MModelTexture, MModelMaterial, MModelCMT, MModelSignature.
	Test method	Visual
	Test type	Conformance
Test 58	/conf/core/mmodel-naming/mmodelgeometry-name	
	Requirement	/req/core/mmodelgeometry-name
	Test purpose	Verify that the names of all MModelGeometry files adhere to the following naming convention: D600_Snnn_Tnnn_MMDC.ft.
	Test method	Visual
	Test type	Conformance
Test 59	/conf/core/mmodel-naming/mmodeldescriptor-name	
	Requirement	/req/core/mmodeldescriptor-name
	Test purpose	Verify that the MModelDescriptor dataset is assigned dataset code 603 and the names of all MModelDescriptor files adhere to the following naming convention: D603_S001_T001_MMDC.xml.
	Test method	Visual
	Test type	Conformance
Test 60	/conf/core/mmodel-naming/mmodeltexture-name	
	Requirement	/req/core/mmodeltexture-name
	Test purpose	Verify that the names of all MModelTexture files adhere to the following naming convention: D601_Snnn_Tnnn_Wnn_TNAM.rgb.
	Test method	Visual
	Test type	Conformance
Test 61	/conf/core/mmodel-naming/mmodelmaterial-name	
	Requirement	/req/core/mmodelmaterial-name
	Test purpose	Verify that the MModelMaterial dataset is assigned dataset code 604 and the names of all MModelMaterial files adhere to the following naming convention: D604_Snnn_Tnnn_Wnn_TNAM.tif.
	Test method	Visual

	Test type	Conformance
Test 62	/conf/core/mmodel-naming/mmodelcmt-name	
	Requirement	/req/core/mmodelcmt-name
	Test purpose	Verify that the MModelCMT dataset is assigned dataset code 605 and the names of all MModelCMT files adhere to the following naming convention: D605_S001_T001_TNAM.xml.
	Test method	Visual
	Test type	Conformance
Test 63	/conf/core/mmodel-naming/mmodelsignature-name	
	Requirement	/req/core/mmodelsignature-name
	Test purpose	Verify that the names of all MModelSignature files adhere to the following naming convention: D606_Snnn_Tnnn_LOD_MMDC.xxx where xxx is the file extension for each file or table required for the Model Signature file
	Test method	Visual
	Test type	Conformance

A.1.16 Tiled Datasets

The following conformance class is designed to determine if the tile dataset have the necessary requirements.

Conformance Class	/conf/core/tiled-data	
Requirements	/req/core/tiled-data	
Dependency	Various XML schema	
Test 64	/conf/core/tiled-data/vector-dataset-limit	
	Requirement	/req/core/vector-dataset-limit
	Test purpose	Verify that the \CDB\MModel\ folder is the root directory of the MModel library which is composed of the following datasets: MModelGeometry, MModelDescriptor, MModelTexture,

		MModelMaterial, MModelCMT, MModelSignature.
	Test method	Visual
	Test type	Conformance
Test 65	/conf/core/tiled-data/latitude-directory-name	
	Requirement	/req/core/latitude-directory-name
	Test purpose	Verify that the Latitude Directory naming is based on a specific policy.
	Test method	Visual
	Test type	Conformance
Test 66	/conf/core/tiled-data/longitude-directory-name	
	Requirement	/req/core/longitude-directory-name
	Test purpose	Verify that the Longitude Directory naming is based on a specific policy.
	Test method	Visual
	Test type	Conformance
Test 67	/conf/core/tiled-data/uref-directory-name	
	Requirement	/req/core/uref-directory-name
	Test purpose	All files stored in the UREF subdirectory of section 3.6.2.5 have the following naming convention: LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn.xxx
	Test method	Visual
	Test type	Conformance

A.1.17 Archive Names

The following conformance class is designed to determine if the archive names have the necessary requirements.

Conformance Class	/conf/core/tiled-data
Requirements	/req/core/archive
Dependency	Various XML schema

Test 68	/conf/core/core/archive/archive-names	
	Requirement	/req/core/archive-names
	Test purpose	Verify that the files inside those archives follow the naming conventions defined here: LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_"extra_tokens".xxx
	Test method	Visual
	Test type	Conformance
Test 69	/conf/core/archive/gsmodegeometry-archive-name	
	Requirement	/req/core/gsmodegeometry-archive-name
	Test purpose	Verify that the files from the GSModelGeometry and GSModelInteriorGeometry datasets have the following naming convention: LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_FC_FSC_MODL.ftt
	Test method	Visual
	Test type	Conformance
Test 70	/conf/core/archive/gsmodeltexture-archive-name	
	Requirement	/req/core/gsmodeltexture-archive-name
	Test purpose	Verify that the files from the GSModelTexture and GSModelInteriorTexture datasets have the following naming convention: LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_TNAM.rgb
	Test method	Visual
	Test type	Conformance
Test 71	/conf/core/archive/msmodelmaterial-archive-name	
	Requirement	/req/core/msmodelmaterial-archive-name
	Test purpose	Verify that the files from the GSModelMaterial and GSModelInteriorMaterial datasets have the following naming convention: LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_TNAM.tif
	Test method	Visual
	Test type	Conformance
Test 72	/conf/core/archive/gsmodeldescriptor-archive-name	
	Requirement	/req/core/gsmodeldescriptor-archive-name

	Test purpose	Verify that the files from the GSModelGeometry and GSModelInteriorGeometry datasets have the following naming convention: LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_FC_FSC_MODL.ft
	Test method	Visual
	Test type	Conformance

A.1.18 NavData Naming Convention

The following conformance class is designed to determine if each field of the NavData file name is correctly selected based on the table 3-32.

Conformance Class	/conf/core/navdata-naming	
Requirements	/req/core/navdata-naming	
Dependency		
Test 73	Test purpose	Verify if All files of the NavData dataset have the appropriate naming convention.
	Test method	Visually check if if All files of the NavData dataset have the following naming convention: D400_Snnn_Tnnn.dbf and Table 3-32: NavData Naming Convention.
	Test type	Conformance

A.1.19 Metadata Datasets

The following conformance class is designed to determine if all the metadata files are located in the correct folder.

Conformance Class	/conf/core/metadata-datasets	
Requirements	/req/core/metadata-datasets	
Dependency	Various XML schema	
Test 74	/conf/core/metadata-datasets	
	Requirement	/req/core/version

	Test purpose	Verify that each CDB Version have a correct version control file.
	Test method	Compare to verify if the content of the Version.xml is based on a xml schema file.
	Test type	Conformance
Test 75	/conf/core/metadata-datasets/attribute-definition	
	Requirement	/req/core/attribute-definition
	Test purpose	Verify that all attribute elements are correct.
	Test method	Compare attributes elements versus a xml XSD file.
	Test type	Conformance
Test 76	/conf/core/metadata-datasets/attribute-schema-level	
	Requirement	/req/core/attribute-schema-level
	Test purpose	Verify that all schema level elements are correct.
	Test method	Compare schema level elements versus a xml XSD file.
	Test type	Conformance
Test 77	/conf/core/metadata-datasets/attribute-value	
	Requirement	/req/core/attribute-value
	Test purpose	Verify that all attribute values are correct.
	Test method	Compare attributes values versus a xml XSD file.
	Test type	Conformance
Test 78	/conf/core/metadata-datasets/attribute-scaler	
	Requirement	/req/core/attribute-scaler
	Test purpose	Verify that all scalers definitions are correct.
	Test method	Compare scalers definitions a xml XSD file.

	Test type	Conformance
Test 79	/conf/core/metadata-datasets/attribute-units	
	Requirement	/req/core/attribute-units
	Test purpose	Verify that all attributes units are correct.
	Test method	Compare attributes unites versus a xml XSD file.
	Test type	Conformance
Test 80	/conf/core/metadata-datasets/ao1	
	Requirement	/req/core/ao1
	Test purpose	Verify that the angular distance feature is recordable or not.
	Test method	Visual
	Test type	Conformance

A.1.20 Navigation Data

The following conformance class is designed to determine if all the NavData dataset represents are correctly located in the folder and have correct materials. NavData supports several simulation subsystems such as the Instrument Landing System (ILS), Inertial Navigation/Global Positioning System, and Microwave Landing System Communications.

Conformance Class	/conf/core/nav-data	
Requirements	/req/core/nav-data	
Dependency	Various XML schema	
Test 81	/conf/core/nav-data/navigation-file-format	
	Requirement	/req/core/navigation-file-format
	Test purpose	Verify that the Navigation Dataset uses a CDB specified vector data format

	Test method	Visually
	Test type	Conformance
Test 82	/conf/core/nav-data/nav-schema-cs2	
	Requirement	/req/core/nav-schema-cs2
	Test purpose	Verify that CS2 value is correct.
	Test method	Visually check if the value of CS2 is T002 for the schema files,
	Test type	Conformance
Test 83	/conf/core/nav-data/nav-key-datasets	
	Requirement	/req/core/nav-key-datasets
	Test purpose	Verify that for each attribute that has an index key in the schema file, an index Key Dataset is available and For Key Datasets, the Dataset CS2 include the last three digits of the index key from the schema file
	Test method	Visual
	Test type	Conformance
Test 84	/conf/core/nav-data/navdata-component	
	Requirement	/req/core/navdata-component
	Test purpose	Verify that the Airport NavData Component is correct.
	Test method	Visually check if for the Airport NavData Component, there are 4 key datasets (for attributes StoraNumber, Ident, IcaoCode and Country)
	Test type	Conformance

A.1.21 Tiled Raster Datasets

The following conformance class is designed to determine if all the all of the CDB raster datasets have correct grid sampling and structure.

Conformance Class	/conf/core/tiled-raster-datasets-general	
Requirements	/req/core/tiled-raster-datasets-general	
Dependency	Various XML schema	
Test 85	/conf/core/tiled-raster-datasets-general/tile-grid-structure	
	Requirement	/req/core/tile-grid-structure
	Test purpose	Verify that the tile grid structure has correct alignment,.
	Test method	Visually
	Test type	Conformance
Test 86	/conf/core/tiled-raster-datasets-general/tile-implicit-corner-computation	
	Requirement	/req/core/tile-implicit-corner-computation
	Test purpose	Verify the latitude and longitude of an implicit corner data element in a grid are correct based on the equation 5-1.
	Test method	Visually check if the value of CS2 is T002 for the schema files,
	Test type	Conformance
Test 87	/conf/core/tiled-raster-datasets-general/tile-implicit-center-computation	
	Requirement	/req/core/tile-implicit-center-computation
	Test purpose	Verify that the position of an implicit center data element in a tile are acceptable referring to the equation 5.2.
	Test method	Visual
	Test type	Conformance

A.1.22 Tiled Elevation Dataset

The following conformance class is designed to determine if the CDB terrain elevation data elements have the necessary requirements.

Conformance Class	/conf/core/tiled-raster-elevation-terrain
Requirements	/req/core/tiled-raster-elevation-terrain
Dependency	Various XML schema

Test 88	/conf/core/tiled-raster-elevation-terrain/primary-terrain-component	
	Requirement	/req/core/primary-terrain-component
	Test purpose	Verify that the Primary Terrain Elevation component of the Elevation dataset is represented as a 1 or 2-channel TIFF image.
	Test method	Visual
	Test type	Conformance
Test 89	/conf/core/tiled-raster-elevation-terrain/terrain-tiff-channel1	
	Requirement	/req/core/terrain-tiff-channel1
	Test purpose	Verify that the first channel of the TIFF image contains the Elevation component and is represented as a floating-point or signed integer value.
	Test method	Visual
	Test type	Conformance
Test 90	/conf/core/tiled-raster-elevation-terrain/terrain-tiff-channel2	
	Requirement	/req/core/terrain-tiff-channel2
	Test purpose	Verify that the second channel, third and fourth channels of the TIFF image contains the necessary components and <i>are</i> stored in correct format.
	Test method	Visual
	Test type	Conformance
Test 91	/conf/core/tiled-raster-elevation-terrain/terrain-hypsography-offline	
	Requirement	/req/core/terrain-hypsography-offline
	Test purpose	Verify that after terrain constraining is performed off-line, hypsography features have AHGT set to True, thereby instructing the SE Tools to constrain the terrain elevation using the supplied coordinates; and, the vector feature <i>is</i> a type PointZ, a MultiPointZ, a PolyLineZ, a PolygonZ or a MultiPatch.
	Test method	Visual
	Test type	Conformance
Test 92	/conf/core/tiled-raster-elevation-terrain/terrain-online-terrain-constraints	
	Requirement	/req/core/terrain-online-terrain-constraints

	Test purpose	Verify that for the Feature's AHGT attribute <i>is</i> set to TRUE in some specific cases listed in section 5.6.1.5
	Test method	Visual
	Test type	Conformance
Test 93	/conf/core/tiled-raster-elevation-terrain/terrain-lpn-use	
	Requirement	/req/core/terrain-lpn-use
	Test purpose	Verify that in the case where features overlap one other, client-devices use the mandatory Layer Priority Number (LPN) attribute.
	Test method	Visual
	Test type	Conformance
Test 94	/conf/core/tiled-raster-elevation-terrain/min-max-elevation-data-type	
	Requirement	/req/core/min-max-elevation-data-type
	Test purpose	Verify that the MinElevation and MaxElevation components <i>are</i> represented correctly according to the formulamentioned in Req 94.
	Test method	Visual
	Test type	Conformance
Test 95	/conf/core/tiled-raster-elevation-terrain/maxculture-data-type	
	Requirement	/req/core/maxculture-data-type
	Test purpose	Verify that the MaxCulture component is represented correctly according to the formulamentioned in Req 95.
	Test method	Visual
	Test type	Conformance

A.1.23 Tiled Terrain Bathymetry

The following conformance class is designed to determine if the bathymetry component consists of all the necessary formats and specifications.

Conformance Class	/conf/core/tiled-terrain-bathymetry	
Requirements	/req/core/tiled-terrain-bathymetry	
Dependency	Various XML schema	
Test 96	/conf/core/tiled-terrain-bathymetry/bathymetry-data-type	
	Requirement	/req/core/bathymetry-data-type
	Test purpose	Verify that the Subordinate Bathymetry component of the Elevation dataset is represented based on the appropriate format, grid size and values.
	Test method	Visually
	Test type	Conformance
Test 97	/conf/core/tiled-terrain-bathymetry/subordinate-alternate-bathymetry-data-type	
	Requirement	/req/core/subordinate-alternate-bathymetry-data-type
	Test purpose	Verify that the first, second, third and fourth channel of the TIFF image are represented correctly based on the appropriate format, grid size and values.
	Test method	Visually check if the value of CS2 is T002 for the schema files,
	Test type	Conformance
Test 98	/conf/core/tiled-terrain-bathymetry/tide-component-data-type	
	Requirement	/req/core/tide-component-data-type
	Test purpose	Verify that the Tide components are represented correctly based on the appropriate format, grid size and values.
	Test method	Visual
	Test type	Conformance

A.1.24 Tiled JPEG Metadata

The following conformance class is designed to determine if the JPEG 2000 files has all the necessary formats and specifications.

Conformance Class	/conf/core/tiled-raster-jpeg-metadata	
Requirements	/req/core/tiled-raster-jpeg-metadata	
Dependency	Various XML schema	
Test 99	/conf/core/tiled-raster-jpeg-metadata/jpeg-origin-metadata	
	Requirement	/req/core/jpeg-origin-metadata
	Test purpose	Verify that the metadata about the origin of data implemented with a JPEG image in the CDB data store is based on the table 5.6.2.1.1.1.
	Test method	Visual
	Test type	Conformance
Test 100	/conf/core/tiled-raster-jpeg-metadata/jpeg-security-metadata	
	Requirement	/req/core/jpeg-security-metadata
	Test purpose	Verify that the metadata about the security of data implemented with a JPEG image in the CDB data store is based on the table 5.6.2.1.1.1.
	Test method	Visual
	Test type	Conformance

A.1.25 Visible Spectrum Terrain Imagery (VSTI)

The following conformance class is designed to determine if the VSTI dataset implicitly follows the grid element conventions and specifications.

Conformance Class	/conf/core/tiled-raster-vsti	
Requirements	/req/core/tiled-raster-vsti	
Dependency	Various XML schema, JPEG 2000 format	
Test 101	/conf/core/tiled-raster-vsti/vsti-component-data-type	
	Requirement	/req/core/vsti-component-data-type

	Test purpose	Verify that the VSTI component is represented as single-channel gray-scale images, or as triple-channel non-paletted color images in JPEG 2000 format. Also verify that no transparency on terrain imagery is performed.
	Test method	Visual
	Test type	Conformance
Test 102	/conf/core/tiled-raster-vsti/client-vsti-texture-selection-rules	
	Requirement	/req/core/client-vsti-texture-selection-rules
	Test purpose	Verify that the Simulation client-devices select the VSTI texture that best represents the simulation data based on the conventions of Req. 102.
	Test method	Visual
	Test type	Conformance

A.1.26 Visible Spectrum Terrain Light Map (VSTLM)

The following conformance class is designed to determine if the VSTI dataset implicitly follows the grid element conventions and specifications.

Conformance Class	/conf/core/tiled-raster-vstlm	
Requirements	/req/core/tiled-raster-vstlm	
Dependency	Various XML schema, JPEG 2000	
Test 103	Test purpose	Verify that the VSTLM component is represented as single-channel gray-scale images, or as triple-channel color images and stored in JPEG 2000 format. Also, if it is a monochrome VSTLM, the implied chrominance of the VSTLM is white.
	Test method	Visually pass if VSTLM component is represented as single-channel gray-scale images, or as triple-channel color images and stored in JPEG 2000 format and if a monochrome VSTLM exists, the implied chrominance of the VSTLM is white.
	Test type	Conformance

A.1.27 Raster Composite Material

The following conformance class is designed to determine if the raster composite material dataset which consist of several (up to 255) composite materials for each pixel follows the necessary conventions and specifications.

Conformance Class	/conf/core/tiled-raster-composite-material	
Requirements	/req/core/tiled-raster-composite-material	
Dependency	Various XML schema	
Test 104	/conf/core/tiled-raster-composite-material/	
	Requirement	/req/core/material-layer-data-type
	Test purpose	Verify that each material layer components is represented as single-channel, material coded one byte unsigned integer value images stored in TIFF.
	Test method	Visual
	Test type	Conformance
Test 105	/conf/core/tiled-raster-composite-material/material-mixture-data-type	
	Requirement	/req/core/material-mixture-data-type
	Test purpose	Verify that each material mixture components is stored in a single-channel TIFF file and all values in the file range from 0.0 (0%) to 1.0 (100%).
	Test method	Visual
	Test type	Conformance
Test 106	/conf/core/tiled-raster-composite-material/tcmt-data-type	
	Requirement	/req/core/tcmt-data-type
	Test purpose	Verify that Terrain Composite Materials Table (TCMT) follows the syntax described in Section 2.5.2.2, Composite Material Tables (CMT).
	Test method	Visual
	Test type	Conformance

A.1.28 Tiled Vector Datasets

The following conformance class is designed to determine if the tiled vector datasets follows the necessary specifications.

Conformance Class	/conf/core/tiled-vector-datasets
Requirements	/req/core/tiled-vector-datasets
Dependency	Various XML schema, CRS

Test 107	/conf/core/tiled-vector-datasets/vector-rule	
	Requirement	/req/core/vector-shp-rule
	Test purpose	Verify that all instances of the feature are saved in the same vector data type in such a way that there is a maximum of one type for lineal features and a maximum of one type for polygon features for each tile (for a maximum of 3 feature vector data files per tile).
	Test method	Visual
	Test type	Conformance
Test 108	/conf/core/tiled-vector-datasets/vector-client-origin-with-model	
	Requirement	/req/core/vector-client-origin-with-model
	Test purpose	Verify that client-devices position the models associated with a feature, and a point feature at the specified coordinate and orientation.
	Test method	In the case of models, visually investigate if the model's coordinate and orientation are correct by checking if the origin is located at a specified coordinate, oriented in accordance to the AO1 attribute, and align the model's Z-axis so that it points up and Z component is positioned to the value specified by the underlying terrain offset by the Z component value if AHGT is not false. And in the case of point feature, visually investigate if the point's coordinate are correct by checking if the origin is located at a specified coordinate, and Z component is positioned to the value specified by the underlying terrain offset by the Z component value if AHGT is not false.
	Test type	Conformance
Test 109	/conf/core/tiled-vector-datasets/model-light-point-position	
	Requirement	/req/core/model-light-point-position
	Test purpose	Verify that the position of light points is expressed using WGS-84 geographic coordinates and the client-device position the center of the light point at the specified coordinate, orient directional light points in accordance to the AO1 attribute.
	Test method	Visual
	Test type	Conformance
Test 110	/conf/core/tiled-vector-datasets/light-point-with-z-position	
	Requirement	/req/core/light-point-with-z-position
	Test purpose	Verify that the client-devices position the light point's center as per the AHGT value.
	Test method	Visually investigate if AHGT is true, the light point's center is positioned to the value specified by the Z component (Absolute Terrain Altitude), irrelevant of the terrain elevation dataset and if

		AHGT is false or not present, the light point's center is positioned to the value specified by the underlying terrain offset by the Z component value.
	Test type	Conformance
Test 111	/conf/core/tiled-vector-datasets/vertex-position	
	Requirement	/req/core/vertex-position
	Test purpose	Verify that the position of vertices is expressed using WGS-84 geographic coordinates and with the correct Z.
	Test method	Visually investigate if the position of vertices is expressed using WGS-84 and the Z component follow these rules: In the case of Shape types that do not have a Z component value, the vertex's height value is referenced to the underlying terrain. For Shape types with a Z component, client-devices position the vertex as per the AHGT value. If AHGT is true, the vertex is positioned to the value specified by the Z component (Absolute Terrain Altitude), irrelevant of the terrain elevation dataset. If AHGT is false or not present, the vertex is positioned to the value specified by the underlying terrain offset by the Z component value.
	Test type	Conformance

A.1.29 Vector Datasets Mandatory Attribute Usage

The following conformance class is designed to determine if the vector datasets attribute follows the necessary specifications.

Conformance Class	/conf/core/tiled-vector-datasets-mandatory-attributes	
Requirements	/req/core/tiled-vector-datasets-mandatory-attributes	
Dependency	Various XML schema	
Test 113	/conf/core/tiled-vector-datasets-mandatory-attributes/mandatory-attribute-compliance	
	Requirement	/req/core/mandatory-attribute-compliance
	Test purpose	Verify that any CDB compliant dataset has no missing mandatory attributes.
	Test method	Visually investigate if all of the mandatory attributes in a CDB dataset are available and have value.
	Test type	Conformance
Test 114	/conf/core/tiled-vector-datasets-mandatory-attributes/optional-attribute-compliance	
	Requirement	/req/core/optional-attribute-compliance

	Test purpose	Verify that a CDB dataset with missing optional attributes <i>is</i> considered compliant although the performance of one or more of the client-devices may be degraded.
	Test method	Visual
	Test type	Conformance
Test 115	/conf/core/tiled-vector-datasets-mandatory-attributes/dataset-instance-schema	
	Requirement	/req/core/dataset-instance-schema
	Test purpose	Verify that: - All the attributes and their values are specified as attribution columns in the instance-level *.dbf file that accompanies the vector dataset. - This file or table is referred to as the Dataset Instance-level file. - Each attribute is uniquely defined by an attribute identifier that is a “case-sensitive” character string of 10 characters or less.
	Test method	Visual
	Test type	Conformance
Test 116	/conf/core/tiled-vector-datasets-mandatory-attributes/attributes-metadata	
	Requirement	/req/core/attributes-metadata
	Test purpose	Verify that the CDB_Attributes.xml metadata file <i>is</i> included in the CDB folder hierarchy under the CDB Metadata directory and is based on a *.xsd schema file that governs the syntax and structure of the attribute metadata file and used to describe all the CDB attributes listed in CDB Attribution.
	Test method	Visual
	Test type	Conformance

A.1.30 Vector Datasets Topology

The following conformance class is designed to determine if the vector dataset topology follows the necessary specifications.

Conformance Class	/conf/core/tiled-vector-datasets-topology	
Requirements	/req/core/tiled-vector-datasets-topology	
Dependency	Various XML schema	
Test 117	/conf/core/tiled-vector-datasets-topology/topology-attributes	
	Requirement	/req/core/topology-attributes
	Test purpose	Verify that for all topological network datasets, the SJID, EJID or

		JID attributes are specified, therefore the features are connected.
	Test method	Pass if all of the the SJID, EJID or JID attributes are not blank.
	Test type	Conformance
Test 118	/conf/core/tiled-vector-datasets-topology/topo-tile-clip	
	Requirement	/req/core/topo-tile-clip
	Test purpose	Verify that tile boundaries of a lineal feature (Network topology) is connected.
	Test method	Pass if the clipping point of a lineal feature in tile boundaries share the same junction identifier (JID) in both tiles.
	Test type	Conformance
Test 119	/conf/core/tiled-vector-datasets-topology/topo-jid-range	
	Requirement	/req/core/topo-jid-range
	Test purpose	Verify that there is a unique identifier at the geocell boundary for the clipping points when the modified or added features overlap two or more geocells.
	Test method	Pass if all SJID, EJID and JID have unique values for all network datasets within the same geocell.
	Test type	Conformance

A.1.31 Elevation Constraints

The following conformance class is designed to determine if the modeler edits and regenerate the elevation datasets based on the elevation constraints and their specifications.

Conformance Class	/conf/core/tiled-elevation-constraints	
Test 121	/conf/core/tiled-elevation-constraints/elevation-constraint-ahgt	
	Requirement	/req/core/elevation-constraint-ahgt
	Test purpose	Verify that the Constraint Features component has the required attributes as mentioned in the Req. 121.
	Test method	Pass if in the case of PointZ, MultiPointZ, PolyLineZ, PolygonZ and MultiPatch feature, the AHGT attribute is set to TRUE. Vector feature types implemented as Point, PointM, MultiPoint, MultiPointM, PolyLine, PolyLineM, Polygon and PolygonM are ignored.
	Test type	Conformance
Test 122	/conf/core/tiled-elevation-constraints/hypsography-elevation-offline	

	Requirement	/req/core/hypsography-elevation-offline
	Test purpose	When performed off-line, verify the instructing of SE Tools to constrain the terrain elevation using the supplied (latitude, longitude, and elevation) coordinates.
	Test method	Pass if the the grid is generating during the off-line CDB compilation process and the hypsography features have AHGT set to True.
	Test type	Conformance
Test 123	/conf/core/tiled-elevation-constraints/hypsograph-vector-types	
	Requirement	/req/core/hypsograph-vector-types
	Test purpose	Verify that vector features are of type PointZ, MultiPointZ, PolyLineZ, PolygonZ, or MultiPatch.
	Test method	Visual
	Test type	Conformance

A.1.32 Tiled Road Networks

The following conformance class is designed to determine if the RoadNetwork Dataset is used to specify tiled road networks.

Conformance Class	/conf/core/tiled-road-network	
Requirements	/req/core/tiled-road-network	
Dependency	Various XML schema	
Test 124	Test purpose	Verify that the road networks use RoadNetwork Dataset to specify its components.
	Test method	Visual
	Test type	Conformance

A.1.33 Tiled Railroad Networks

The following conformance class is designed to determine if the RailRoadNetwork Dataset is used to specify tiled railroad networks.

Conformance Class	/conf/core/tiled-railroad-network	
Requirements	/req/core/tiled-railroad-network	

Dependency	Various XML schema	
Test 125	Test purpose	Verify that the railroad networks use RailRoadNetwork Dataset to specify its components.
	Test method	Visual
	Test type	Conformance

A.1.34 Tiled PowerLine Networks

The following conformance class is designed to determine if the PowerLineNetwork Dataset is used to specify tiled powerline networks.

Conformance Class	/conf/core/tiled-powerline-network	
Requirements	/req/core/tiled-powerline-network	
Dependency	Various XML schema	
Test 126	Test purpose	Verify that the powerline networks use PowerLineNetwork Dataset to specify its components.
	Test method	Visual
	Test type	Conformance

A.1.35 Tiled Hydrography Networks

The following conformance class is designed to determine if the HydrographyNetwork Dataset is used to specify tiled hydrography networks.

Conformance Class	/conf/core/tiled-hydro-network	
Requirements	/req/core/tiled-hydro-network	
Dependency	Various XML schema	
Test 127	Test purpose	Verify that the hydrography networks use HydrographyNetwork Dataset to specify its components.
	Test method	Visual
	Test type	Conformance

A.1.36 Vector Composite Material Table (VCMT)

The following conformance class is designed to determine if the CDB geocells have at least one VCMT.xml file with the required list of attributes for the composite materials of vector datasets.

Conformance Class	/conf/core/vcmt	
Requirements	/req/core/vcmt	
Dependency	Various XML schema	
Test 128	Test purpose	Verify that there is on VCMT for each CDB Geocell which provides a list of the Composite Materials shared by all vector datasets.
	Test method	Visually pass if there is on VCMT for each CDB Geocell.
	Test type	Conformance

