Volume 1: OGC CDB Core Standard

Model and Physical Data Store Structure

Table of Contents

Open Geospatial Consortium

Submission Date: 2019-xx-xx

Approval Date: 2020-xx-xx

Publication Date: 2020-xx-xx

External identifier of this OGC® document: http://www.opengis.net/doc/IS/CDB-core/1.2

Additional Formats (informative):

Internal reference number of this OGC® document: 15-113r6

Version: 1.2

Category: OGC® Implementation Standard

Editor: Carl Reed, PhD

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

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:	Candidate
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.

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 simulated[2].

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 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

Organization name(s)

v. Submitters

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

Name	Affiliation

Carl Reed	Carl Reed & Associates
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 CDB Version 1.0, effort was invested to begin aligning terminology and concepts with the OGC standards baseline, specifically in the coordinate reference system discussions and requirements. For version 1.1, the primary effort was focused on resolved several Chnage Requests and adding guidance on incorporating metadata into a CDB dta store.

This version of the CDB standard is backwards compatible with versions 1.0 and 1.1 of the OGC CDB standard.

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 Version 1.0 and Version 1.1 public comment periods, 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 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 may allow the use and implementation of a CDB structured data store for application areas other than aviation simulators.

NOTE: For version 1.2 a major enhancement is the specification of using OGC GeoPackages in a CDB data store.

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.

Chapter 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 formats [3].

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 16 Volumes, one being 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 Rules for Encoding CDB Vector Data using Shapefiles (Best Practice).
- Volume 5: OGC CDB Radar Cross Section (RCS) Models (Best Practice).
- Volume 6: OGC CDB Rules for Encoding CDB Models 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: http://schemas.opengis.net/cdb/ 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 Navaids Attribution and Navaids Attribution Enumeration Values (Best Practice).
- Volume 13: OGC CDB Rules for Encoding CDB Vector Data using GeoPackage (Normative, Optional Extension).
- Volume 14: OGC CDB Guidance on Conversion of CDB Shapefiles into CDB GeoPackages (Best Practice).

- Volume 15: OGC CDB Optional Multi-Spectral Imagery Extension (Normative).

The major enhancements for version 1.2 are the definition of 1.) rules for encoding GeoPackages in a CDB data store, 2.) documenting best practices for conversion of CDB ShapeFiles into CDB GeoPackages, and 3.) resolution and incorporation of changes defined in <X> OGC Change Requests.

The GeoPackage encoding is an optional extension and does not need to be implemented to provide a compliant CDB data store to the community. The only caveat is that if the developer of a version of a CDB data store wishes to use GeoPackages, then that version of the data store must use only the CDB GeoPackage encoding. Shapefiles and GeoPackages cannot be mixed in a unique version of a CDB data store.

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 site[4].

1.2. References

Normative References

Please see the Bibliography in this CDB volume 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>.

1.4. Conventions

This section 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/core/cdb/1.0

An example might be:

req/core/platform

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, controlled vocabularies, to store global datasets, to add attributes and information to 3d models, such as an OpenFlight model, to describe base and composite materials, and so forth.

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

- BaseMaterialTable.xsd
- CompositeMaterialTable.xsd
- Configuration.xsd
- Defaults.xsd
- FeatureDataDictionary.xsd
- Lights.xsd
- LightsTuning.xsd
- ModelComponents.xsd
- ModelMetadata.xsd
- OpenFlightModelExtensions.xsd
- VectorAttributes.xsd
- Version.xsd

Clause 1.4.3 and 1.4.4 provides more information on these files.

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://schemas.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://schemas.opengis.net/cdb/Version/1.1"

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

Name/1.1

in any implementation.

1.4.3. CDB Metadata, Controlled Vocabulary, and Metadata Files

There are a number of CDB XML files that are stored and referenced from the CDB metadata folder. First, some terms are defined

1.4.3.1. Metadata

Metadata is "https://en.wikipedia.org/wiki/Data[data] [information] that provides information about other data”[5]. In the geospatial community and the rapidly emerging spatial web community, metadata is critical to operations such as discovery and determining whether a resource is “fit for purpose”. Three distinct types of metadata exist: **descriptive metadata**, **structural metadata**, and **administrative metadata** [6].

- Descriptive metadata describes a resource for purposes such as discovery and identification. It can include elements such as title, abstract, author, and keywords.
- Structural metadata is metadata about containers of metadata and indicates how compound objects are put together, for example, how pages are ordered to form chapters.
- Administrative metadata provides information to help manage a resource, such as when and how it was created, file type and other technical information, and who can access it. <end Wikipedia>

The geospatial community has a long and extensive history in defining and using metadata for geospatial resources. Without metadata, discovery of required resources and determination of whether a resource is “Fit for Purpose” becomes difficult if not impossible. The geospatial community makes use of all three types of metadata, although the first and third are more critical.

The CDB use of metadata focuses on use cases 1 and 3.

1. Controlled Vocabulary

   Controlled vocabularies provide a way to organize knowledge for subsequent retrieval and use. They are used in subject indexing schemes, subject headings, thesauri,[1][2] taxonomies and other forms of knowledge organization systems. Controlled vocabulary schemes mandate the use of predefined, authorized terms that have been preselected by the designers of the schemes, in contrast to natural language vocabularies, which have no such restriction. The use of controlled vocabularies in standards such as CDB can significantly increase interoperability and consistent understanding of the semantics. Controlled vocabularies typically are managed through formal processes and official governance.

2. Enumerations

   In computer programming, an enumerated type (also called enumeration) is a data type consisting of a set of named values called elements, members, enumeral, or enumerators of the type. The enumerator names are usually identifiers that behave as constants in the language.Similarily, in a database enumerated (enum) types are data types that comprise a static, ordered set of values. They are equivalent to the enum types supported in a number of programming languages. An example of an enum type might be the days of the week, or a set of status values for a piece of data.

1.4.3.2. CDB metadata, controlled vocabularies, and enumerations summary table

The following is a list of these files. While the general term being used is “metadata” in terms of the file system structure, a number of these files are either controlled vocabularies or attribute files. Please read Clauses 1.4.4, 1.5, and 3.1.1 Metadata Directory for more detailed information on the files maintained in that folder. The following table identifies the files stored in the metadata folder and whether they are metadata or controlled vocabularies.

Name	Location	Type[7]	Extension	M/O[8]
CDB_Attributes	\CDB\Metadata	CV	.xml	O
Configuration	\CDB\Metadata	M	.xml	O
Datasets	\CDB\Metadata	CV	.xml	O
Lights	\CDB\Metadata	CV	.xml	O
Lights_*xxx*	\CDB\Metadata	CV	.xml	C
Defaults	\CDB\Metadata	E	.xml	O
Materials	\CDB\Metadata	CV	.xml	O
Modelcomponents	\CDB\Metadata	CV	.xml	O
MovingModelCodes	\CDB\Metadata	E	.xml	O
Version	\CDB\Metadata	M	.xml	M

Name	Location	Type[7]	Extension	M/O[8]
FeatureDataDictionary	\CDB\Metadata	CV	.xml	O
DISCountryCodes	\CDB\Metadata	E	.xml	O
Globalgeometadata	\CDB\Metadata	M	.<ext>[9]	O
Localgeometadata	Determined by directory path rules	M	.<ext>	O

Each of these files is described in detail later in this document.

In CDB version 1.1 and later, additional metadata requirements and elements are specified. These are traditional metadata including geospatial metadata. Specifically, the reader should reference clauses 3.1.1, 3.1.2, and 5.1 (special focus on 5.1.6). Also, make special note of the guidance in clause "3.2.3.2 How to handle the metadata directory."

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 contain enumerations and controlled vocabularies that are necessary to expand the patterns:

- /CDB/Metadata/FeatureDataDictionary.xml provides the list of directory names associated with feature codes.
- /CDB/Metadata/MovingModelCodes.xml provides the list of names for DIS Entity Kinds, Domains, and Categories.
- /CDB/Metadata/DISCountryCodes.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.

The following file extensions are used:

File Format	Minimal Version Number	Extension
TIFF	6.0	*.tif
SGI Image	1.0	*.rgb
JPEG 2000	1.0	*.jp2
OpenFlight	16.0	*.flt
Shapefile	Esri White Paper, July 98	*.shp, *.shx

File Format	Minimal Version Number	Extension
dBASE	III+	*.dbf, *.dbt
XML	1.0 and later	*.xml, *.xsd
ZIP	6.3.1 and later	*.zip
GeoPackage	1.1 and later	*.gpkg

Previous version sf the above table had a column labeled: *CDB Client-device Behavior for Prior Versions"*. All rows had the label *Ignores data*. The column has been removed but the value is still valid.

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 codes [10].

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 inter-connected 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 if 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 Standard.

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

The current CDB standard relies on established industry formats:

- **TIFF**: TIFF encoding rules are defined in Volume 10: OGC CDB Implementation Guidance;
- **OpenFlight**: The Best Practice use of the OpenFlight [11] format (Volume 6: OGC CDB Rules for Encoding Data using OpenFlight);
- **SGI Image**: A native raster graphics file format also known as RGB file format.
- **Shapefile**: 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;
- **GeoPackage**: Using GeoPackages for use in a CDB data store. The GeoPackage content encoding rules are documented in the OGC CDB Optional Extension for Structuring a CDB compliant GeoPackage standard;
- **JPEG 2000**: 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, SGI Image, Shapefile, GeoPackage, and JPEG 2000 formatted files are to be interpreted by CDB-compliant simulator readers.

In addition, a new CDB topic volume for CDB Version 1.2 defines the Best Practices for conversion of CDB Shapefiles into CDB GeoPackages.

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 32 [12]),
- SGI Image File Format (http://paulbourke.net/dataformats/sgirgb/sgiversion.html),
- Table of Dataset Codes (Annex Q [13])
- 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 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; and
- 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 well known and supported data formats endorsed by leaders of the Modeling and Simulation (M&S) 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: SGI Image (RGB).
- For the instancing and attribution of statically positioned point, lineal and polygon 2D/3D culture features: Shapefile or GeoPackage.
- For a representation of terrain raster imagery comprising a well-defined and accepted compression method that allows both lossy and lossless schemes: JPEG 2000.

Note that the OGC CDB Standards Working Group will consider developing best practice guidance for using other industry standard formats and encodings, such as OGC CityGML or OGC InDoorGML.

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 may evolve to enable the use of other international and de-facto geospatial encoding structures.*

The CDB specified storage structure enables 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”. 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-devices [14]. Furthermore, since there is no standardization of the SE representation internal to client-devices (they vary by type [15], 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 *A CDB implementation SHALL include all elements of the CDB Core Data Model as defined in Annex A.*

1.7.1. Platform Requirements

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

This section was moved to section 5.3 Volume 10, Implementation Guidance

1.7.1.5. Literal Case

Requirement 5	http://www.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://www.opengis.net/spec/CDB/1.0/core/raster-imagery-compression JPEG 2000 *SHALL* 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://www.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://www.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://www.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://www.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 ±90° and ±180° 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.

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

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

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

[4] www.opengeospatial.org/cite

[5] http://www.merriam-webster.com/dictionary/metadata

[6] National Information Standards Organization (NISO)

[7] CV = Controlled Vocabulary, M = Metadata, E = Enumeration

[8] M = Mandatory, O = Optional, C = Conditional

[9] <ext> could be xml for XML, jsn for JSON, and other extensions based on the encoding technology used for the geospatial metadata

[10] Currently, these are a mixture of FACC, SEDRIS, DGIWG, DO-272B, UHRB, and other codes, Future revisions of the CDB standard shall provide guidance on using other feature code lists.

[11] 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.

[12] Annex O can be found in Volume 2 CDB Core Model and Physical Structure Annexes

[13] OGC CDB Core: Model and Physical Structure: Informative Annexes

[14] Example client devices are: UAV simulators, mission planning simulators, , helicopter training simulators, and so fort

[15] It is precisely the intent of the CDB standard to provide such standardization.

Chapter 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 Abstract Specification (AS). From the DGGS AS:

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

For performance, a CDB data store is tiled. Both raster and vector-based data sets are tiled. 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-seconds 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-seconds 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 distances [16].

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° × 1°, in zone 4 and 6 which goes from latitude 50° to 70° the cell size is 1° × 2°. 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 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://www.opengis.net/spec/CDB/1.0/core/geocell-extent A CDB Geocell *SHALL* start and end on a whole degree along the longitudinal axis.

Geocell length

Requirement 12	http://www.opengis.net/spec/CDB/1.0/core/geocell-length The length of the CDB Geocell *SHALL* 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.

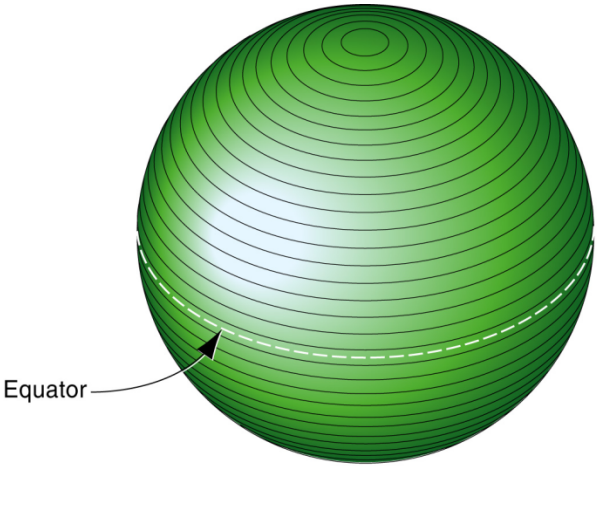

Figure 2-1. CDB Earth Slice Representation

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 latitude and where the *y*-axis is always pointing north, aligned with WGS-84 lines of longitude. 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° meridian [17]. 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.

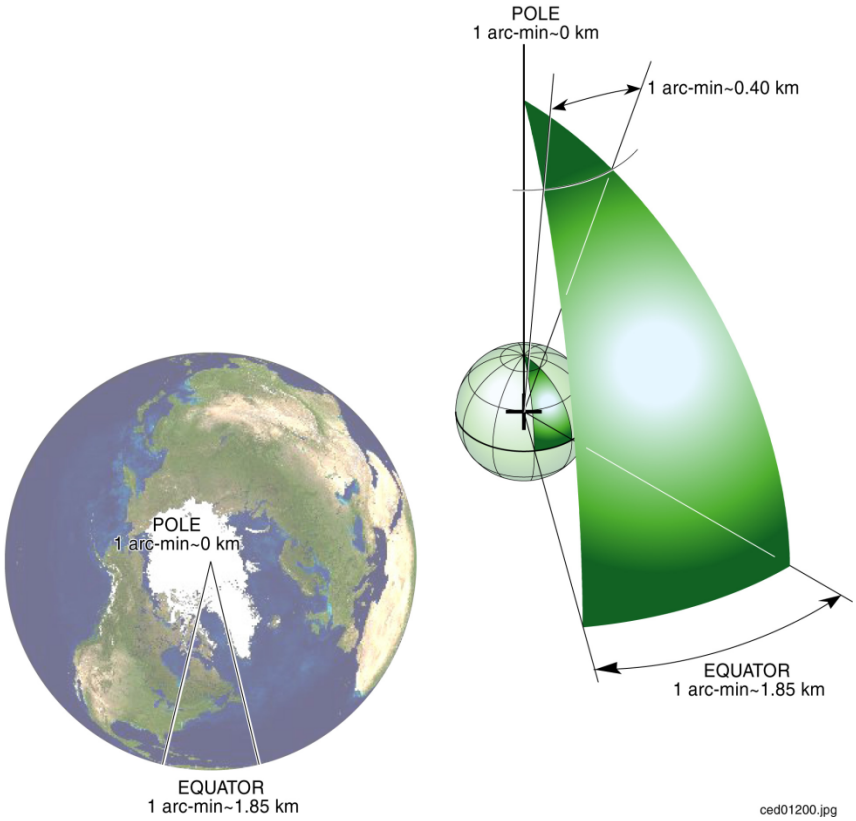

Figure 2-2. Variation of Longitudinal Dimensions version Latitude

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 (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 (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 (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).

$$ArcSecLatUnitInCDBGeoCell = \frac{180 \text{ degrees} \times 3600 \text{ arcsec/degree}}{NbYUnitInCDBGeoCell \times NbEarthSlice}$$

$$ArcSecLatUnitInCDBGeoCell = \frac{180 \text{ degrees} \times 3600 \text{ arcsec/degree}}{1024 \frac{\text{unit}}{\text{slice}} \times 180 \text{ slices}}$$

$$ArcSecLatUnitInCDBGeoCell = 3.515625 \text{ arcsec/unit}$$

Equation (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).

$$XUnit_{LOD} = \frac{XUnit}{2^{LOD}}$$

$$YUnit_{LOD} = \frac{YUnit}{2^{LOD}}$$

Equation (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).

$$NbXUnitInCDBGeocell_{LOD} = NbXUnitInCDBGeocell \times 2^{LOD}$$
$$NbYUnitInCDBGeocell_{LOD} = NbYUnitInCDBGeocell \times 2^{LOD}$$
$$TotalNbUnitInCDBGeocell_{LOD} = NbXUnitInCDBGeocell_{LOD} \times NbYUnitInCDBGeocell_{LOD}$$

Equation (2–6)

Requirement 13	http://www.opengis.net/spec/CDB/1.0/core/tile-sizes Tile sizes *SHALL* 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 km × 111 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 mm × 13 mm with a corresponding grid spacing of about 13 µm.

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.

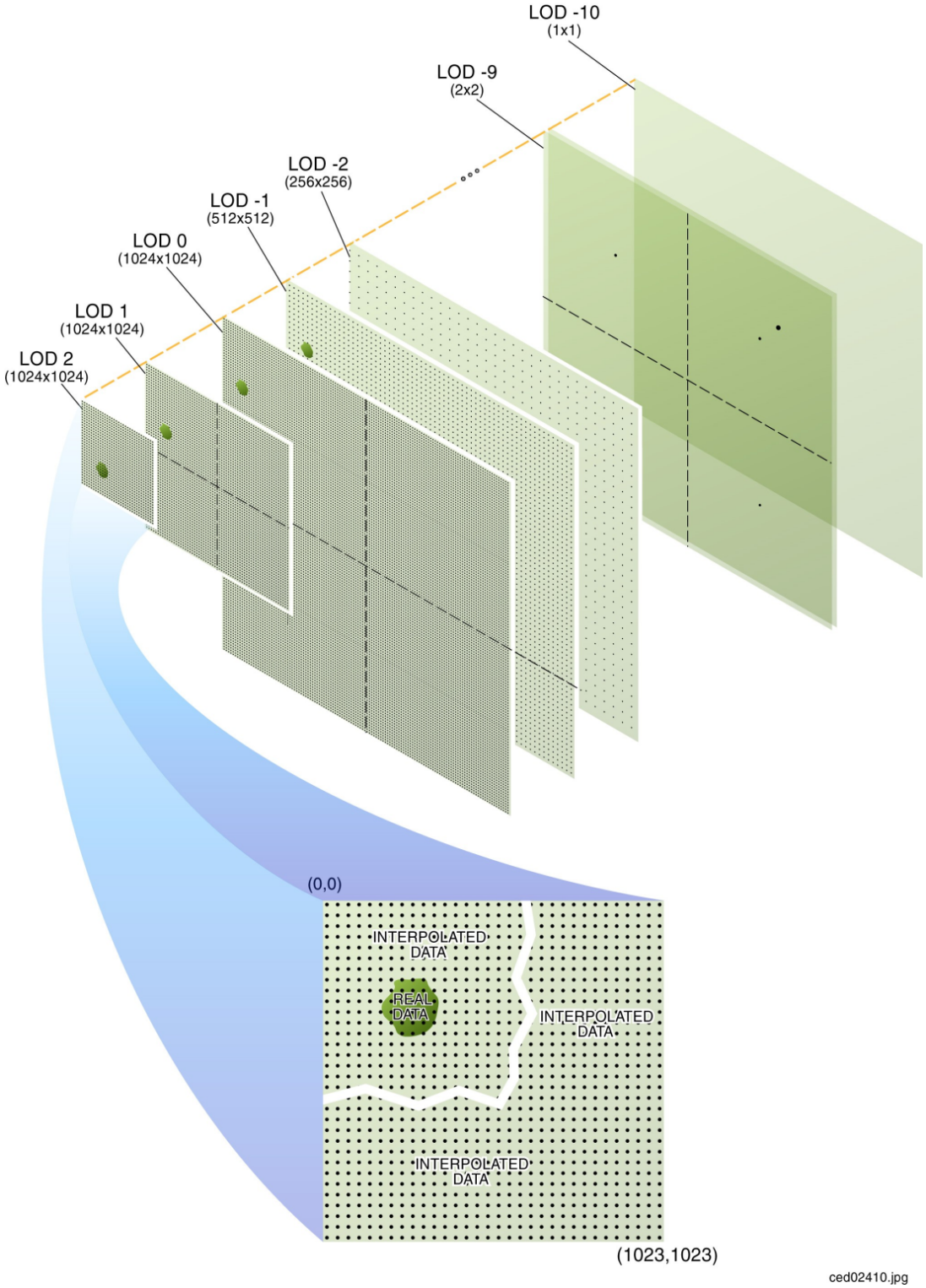

Figure 2-3. Tile-LOD Hierarchy

2.1.2.1. Tile LoD Area Coverage Rules

Requirement 14	http://www.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 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 courser tile exists at LOD *n*−1 covering the area of the tile at LOD *n*, until LOD −10 is reached.

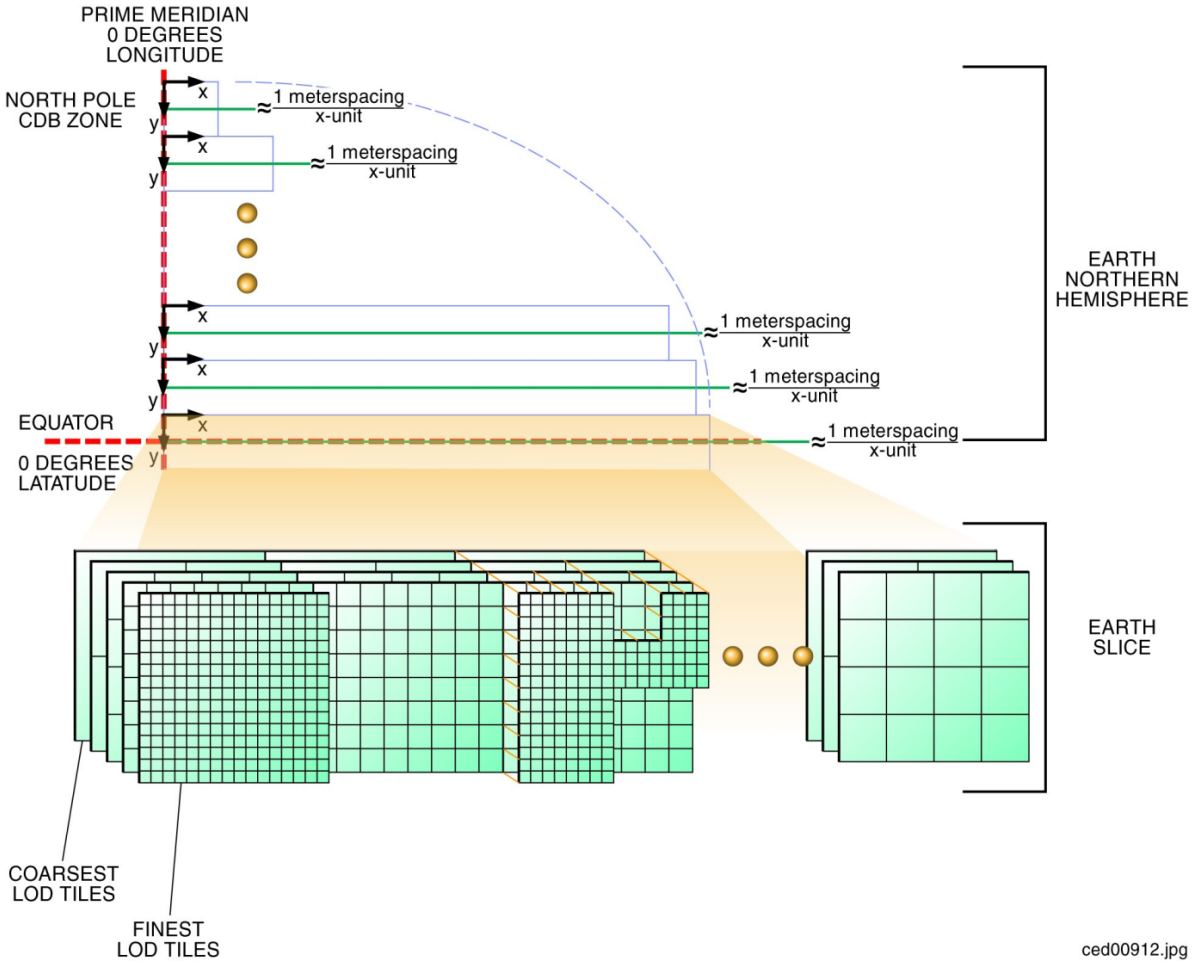

Figure 2-4. Earth Slice Example (Five Levels of Detail)

2.1.3. Tile-LOD Hierarchy Rules

Requirement 15	http://www.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.

2.1.4. Tile-LOD Replacement Rules

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

Requirement 16	http://www.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.

2.1.5. 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 2-3: 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 +89° 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° 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

Please refer to Clause 5 in Volume 10: OGC CDB Implementation Guidance (Best Practice).

2.3. Light Naming

The CDB standard defines rules that unambiguously tag any modeled light point [18] 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, polygons [19], 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 that 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 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://www.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 backslash[20] “\” 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 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 Components [21] 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 representthe following.

- **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
- **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://www.opengis.net/spec/CDB/1.0/core/composite-materials-resolution All references to Composite Materials *SHALL*, 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://www.opengis.net/spec/CDB/1.0/core/base-material-name The Base Material name *SHALL* 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 *SHALL* always begin with the "BM_" string, followed by a unique arbitrary string that respects the following conventions: * 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%** [22]**.** 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.

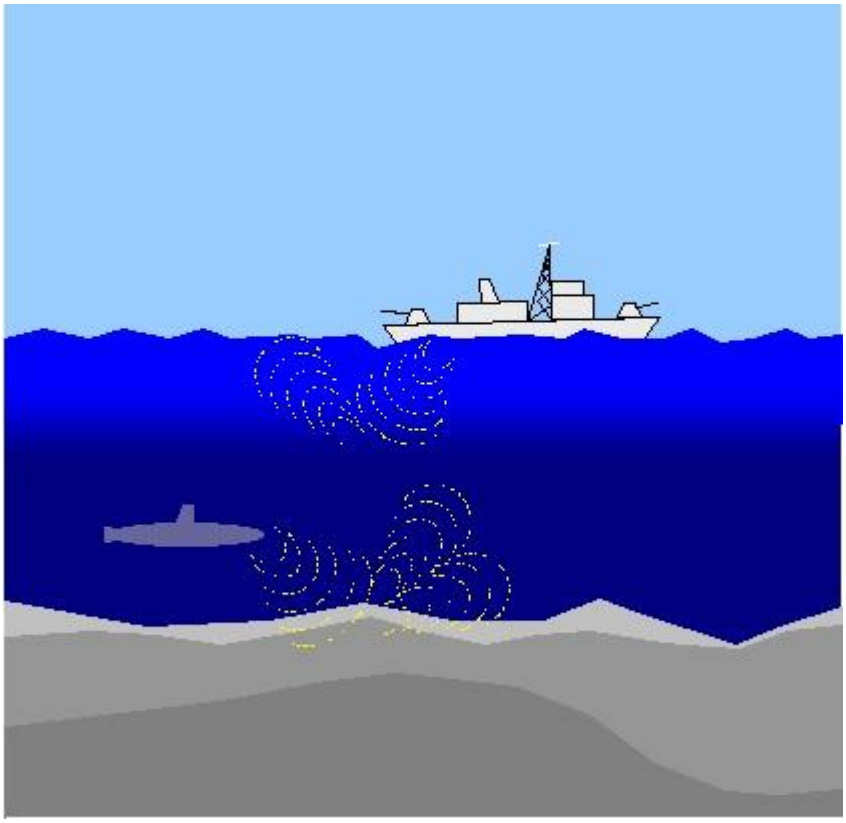

Figure 2-6. Seabed Composite Material

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

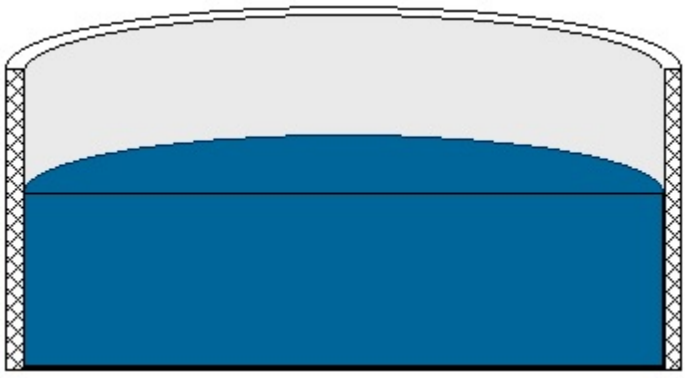

Figure 2-7. Oil Tank Composite Materials

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).

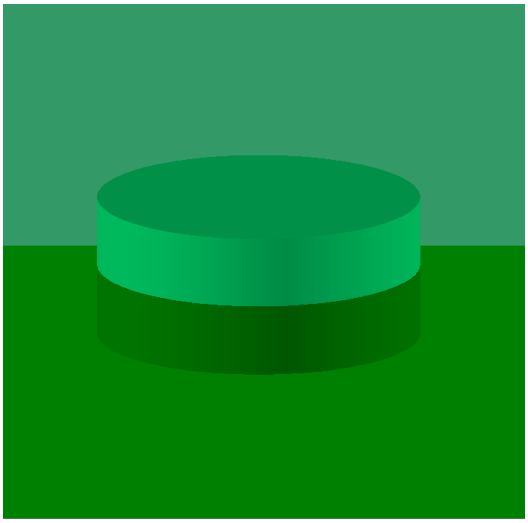	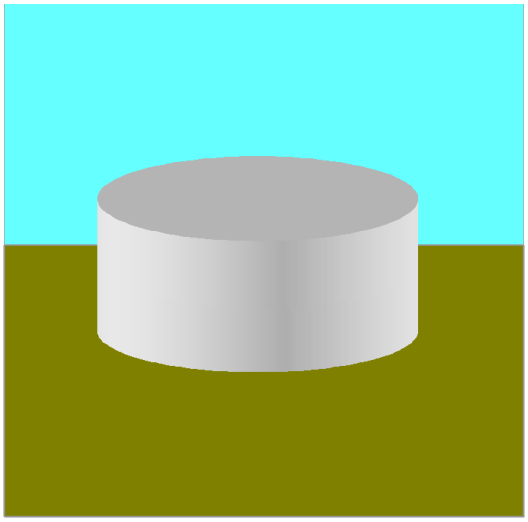
Thermal Infrared	Visible

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://www.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://www.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://www.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.

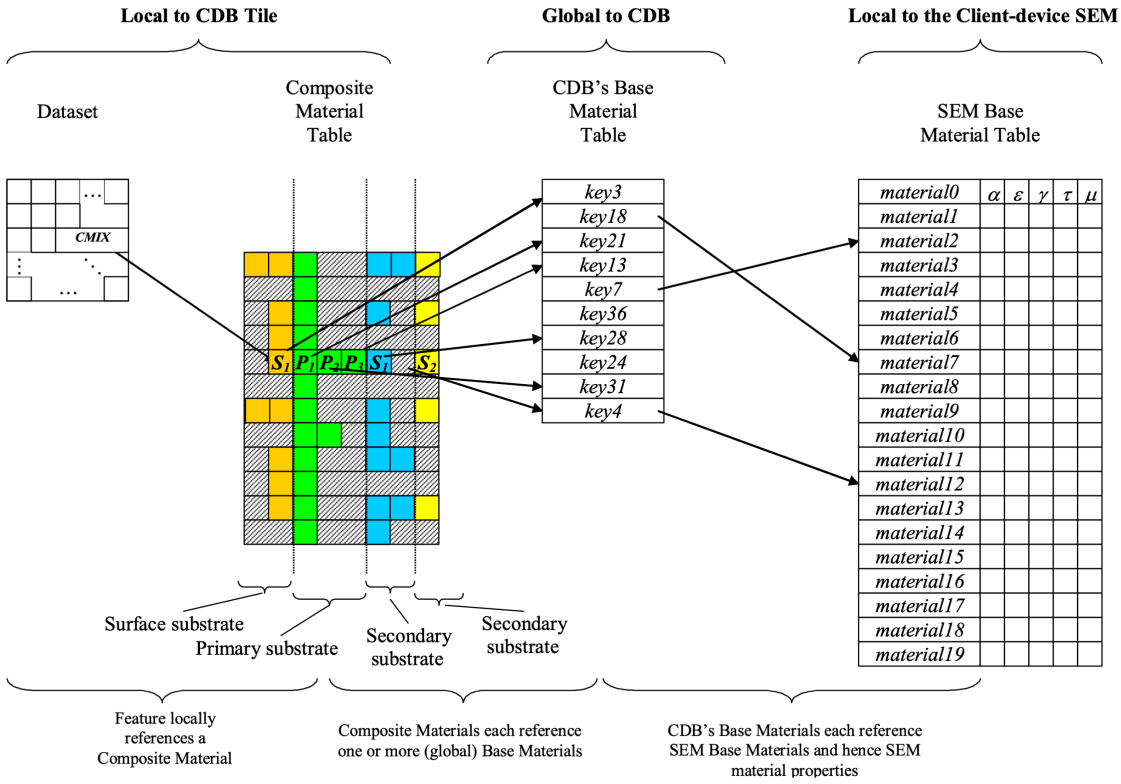

Figure 2-9. Flow of Material Attribution Data

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://www.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://www.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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

Chapter 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 *SHALL* be under the root directory or within a subdirectory under the root directory
Requirement 30	req/core/root-give-path Run-time applications *SHALL* 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 *SHALL* 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 *SHALL* 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\MModel\:** 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.

3.1.1.1. Global_Spatial Metadata File

This file contains metadata, including spatial metadata, pertinent to an entire CDB data store.

Please refer to Clause 5.1.x for detailed information for guidance on metadata elements should be specified for the global metadata elements. The specification of global metadata is mandatory in CDB version 1.1 or later. Please note that this standard does not specify which metadata standard shall be used. Instead, for flexibility and in recognition that different communities use different metadata standards or profiles of metadata standards, the CDB standard provides a list of allowed metadata standards and profiles of those standards. The reader should reference Clause 5.1.6 and Clause 5.1.12 for how the metadata standard used in a CDB data store is specified as part of the "Version" metadata.

3.1.1.2. "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 5.1.3 Light Name Hierarchy Metadata for a description of the light point name hierarchy file.

3.1.1.3. "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 Components [23] is part of the CDB standard distribution package and can be found in the following file: `\CDB\Metadata\Model_Components.xml`. Refer to section 5.1.4 of this document - Model Components Definition Metadata - for a description of the model component file.

3.1.1.4. "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 5.1.5 Base Material Table for a description of the materials definition file.

3.1.1.5. "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 Volume 2"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 5.1.6 Default Values Definition Metadata for a description of the defaults definition file.

3.1.1.6. "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 5.1.8, Version Metadata. See Requirement 38.

3.1.1.7. “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 5.1.9 CDB Attributes Metadata.

3.1.1.8. “Geomatics_Attributes” Metadata file

This optional 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 5.1.10 (this volume), Geomatics Attributes Metadata. 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.

3.1.1.9. “Vendor_Attributes” Metadata file

This optional 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 5.1.11 (this volume), Vendor Attributes Metadata. 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.

3.1.1.10. 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 5.1.3.1, Client Specific Lights Definition Metadata for a description of the client specific lights definition file.

3.1.1.11. “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 5.1.13, Configuration Metadata.

3.1.2. Metadata File Examples

Requirement 33	http://www.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://www.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.

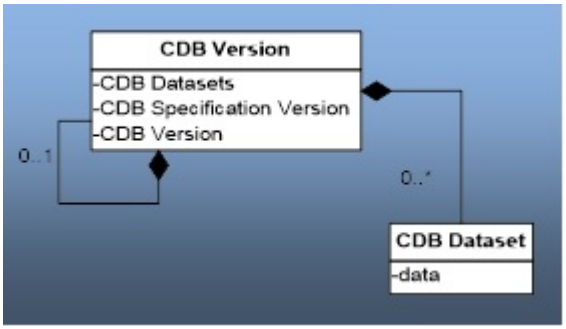

Figure 3-1. UML Diagram of CDB Version Concept

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.

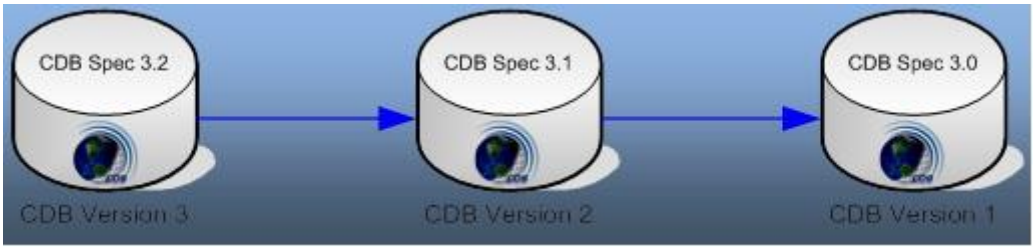

Figure 3-2. A Valid Chain of CDB Versions

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.

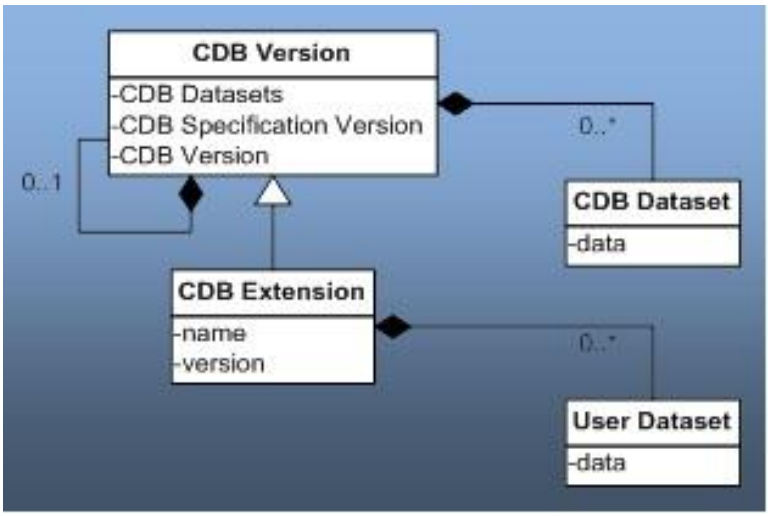

Figure 3-3. UML Diagram of CDB Extension Concept

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 name [24] on any disk device or volume under the target file system it is stored on.

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

Requirement 36	http://www.opengis.net/spec/CDB/1.0/core/cdb-version-path A CDB Version path name *SHALL* 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	http://www.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.

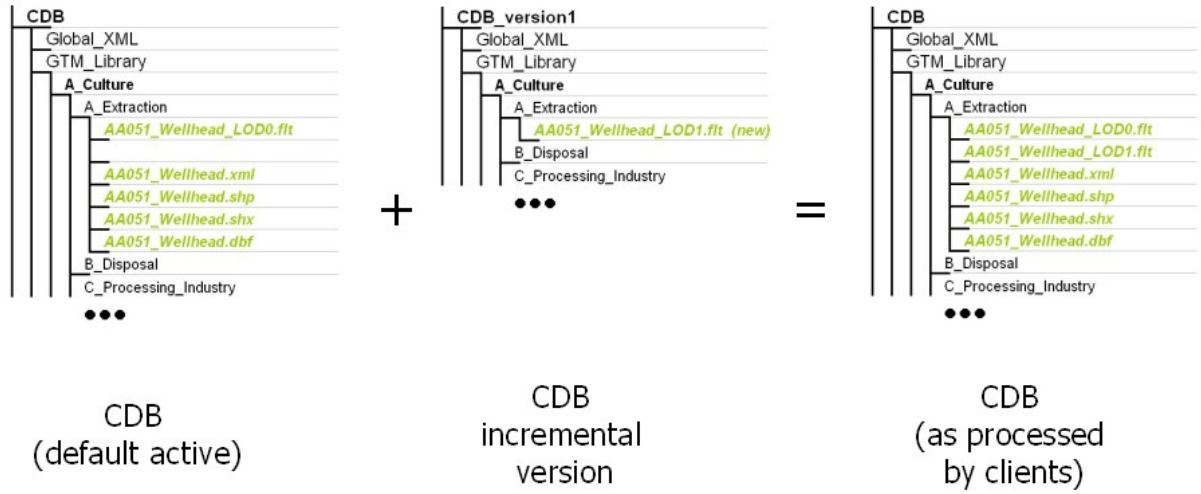

Figure 3-4. Adding Content to the CDB data store

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.

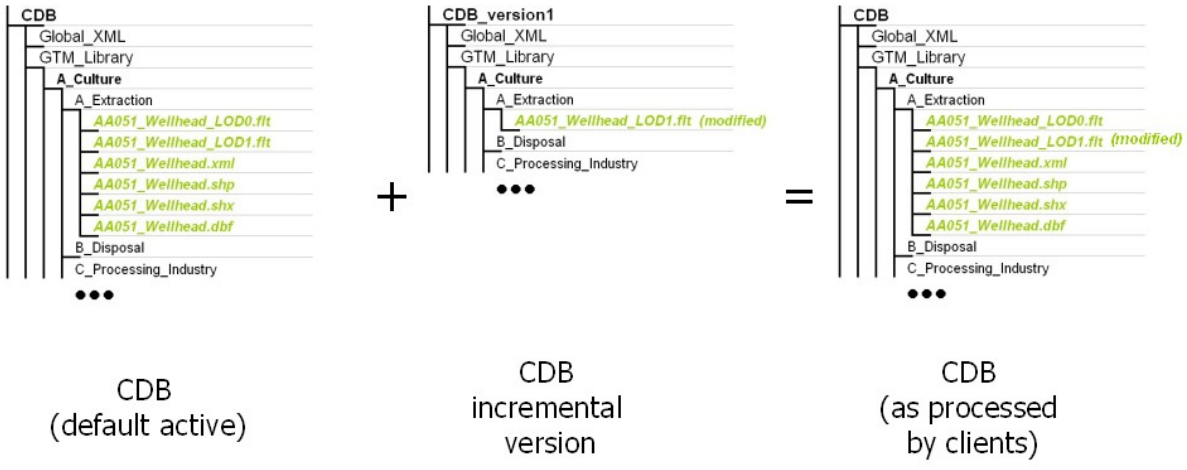

Figure 3-5. Modifying Content of the CDB Data Store

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.

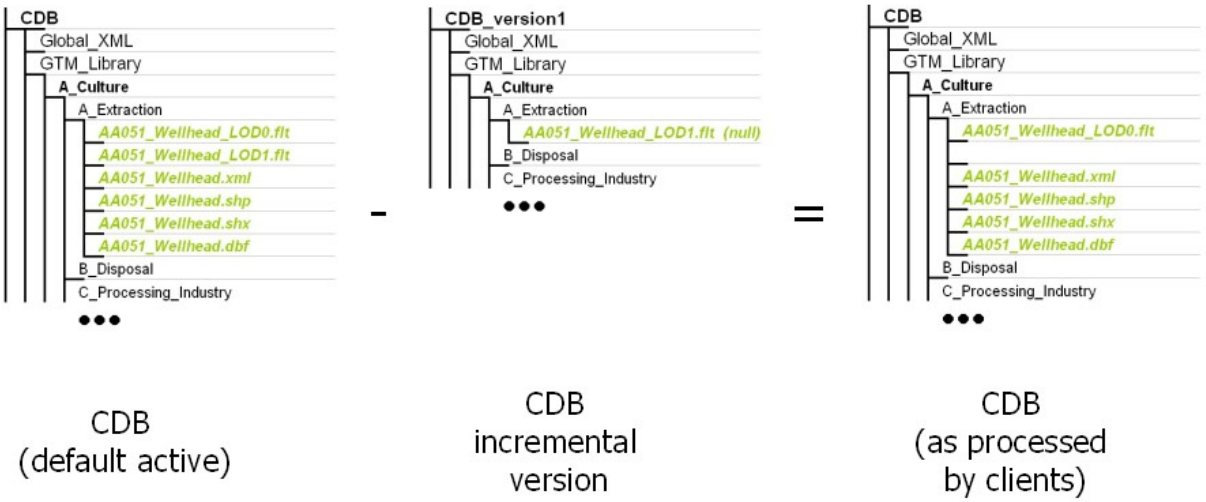

Figure 3-6. Deleting Content from a CDB Data Store

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.

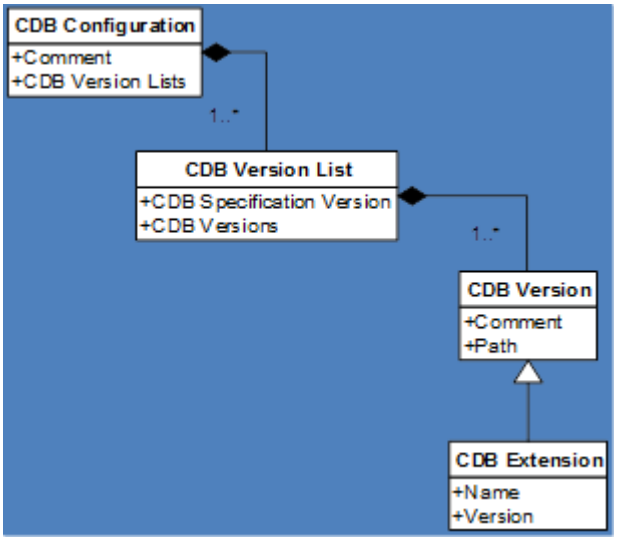

Figure 3-7. UML Diagram of CDB Configuration Concept

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://www.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 versions[25]. 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	http://www.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:

1. Modeled representation of geotypical and geospecific 2D lineal-features such as roads, runways and taxiways, stripes.
2. 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 simulation [26], 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.

When a feature is tagged as geotypical…

1. the modeler is making a statement about his knowledge of the very high probability of repeatedly encountering that model type within the CDB **and…**
2. 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…

1. 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…**
2. 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://www.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://www.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.

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

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

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

In all three equations, the value 111319 represents the approximate length in meters of an arc of one degree at the equator [27].

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 > $5.565950 \times 10^{+4}$	TS > $6.957438 \times 10^{+3}$	BSD > $4.452760 \times 10^{+5}$
-9	SS > $2.782975 \times 10^{+4}$	TS > $3.478719 \times 10^{+3}$	BSD > $2.226380 \times 10^{+5}$
-8	SS > $1.391488 \times 10^{+4}$	TS > $1.739359 \times 10^{+3}$	BSD > $1.113190 \times 10^{+5}$
-7	SS > $6.957438 \times 10^{+3}$	TS > $8.696797 \times 10^{+2}$	BSD > $5.565950 \times 10^{+4}$
-6	SS > $3.478719 \times 10^{+3}$	TS > $4.348398 \times 10^{+2}$	BSD > $2.782975 \times 10^{+4}$
-5	SS > $1.739359 \times 10^{+3}$	TS > $2.174199 \times 10^{+2}$	BSD > $1.391488 \times 10^{+4}$
-4	SS > $8.696797 \times 10^{+2}$	TS > $1.087100 \times 10^{+2}$	BSD > $6.957438 \times 10^{+3}$
-3	SS > $4.348398 \times 10^{+2}$	TS > $5.435498 \times 10^{+1}$	BSD > $3.478719 \times 10^{+3}$
-2	SS > $2.174199 \times 10^{+2}$	TS > $2.717749 \times 10^{+1}$	BSD > $1.739359 \times 10^{+3}$
-1	SS > $1.087100 \times 10^{+2}$	TS > $1.358875 \times 10^{+1}$	BSD > $8.696797 \times 10^{+2}$
0	SS > $5.435498 \times 10^{+1}$	TS > $6.794373 \times 10^{+0}$	BSD > $4.348398 \times 10^{+2}$
1	SS > $2.717749 \times 10^{+1}$	TS > $3.397186 \times 10^{+0}$	BSD > $2.174199 \times 10^{+2}$
2	SS > $1.358875 \times 10^{+1}$	TS > $1.698593 \times 10^{+0}$	BSD > $1.087100 \times 10^{+2}$
3	SS > $6.794373 \times 10^{+0}$	TS > 8.492966×10^{-1}	BSD > $5.435498 \times 10^{+1}$
4	SS > $3.397186 \times 10^{+0}$	TS > 4.246483×10^{-1}	BSD > $2.717749 \times 10^{+1}$
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

GSModel, 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 [28], SEDRIS [29], and UHRB [30] dictionaries. The CDB FDD makes use of feature codes [31] (FC) 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.

Sections 5.7.1.3.24 and 5.7.1.3.25 respectively provide additional information on feature attributes.

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.

3.3.8.2. 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). Featuture 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.

Feature codes are defined and stored in a dictionary of features, attributes and attribute values organized in a standardized coding system [32]. 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.

3.3.8.3. 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.

3.3.8.4. 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 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.

3.3.8.5. Texture Name

Requirement 42	http://www.opengis.net/spec/cdb/1.0/core/texture-name The name of 3D model textures *SHALL* be a character string having a minimum of 2 characters and a maximum length of 32 characters. The first two characters *SHALL* 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 *SHALL* represents the texture name and is used to compose texture file and directory names. The following directory structure *SHALL* be used by CDB Model texture-related datasets: \A\B\TNAM{set:cellbgcolor:#FFFFFF}

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:

```
\H\O\house\
```

3.3.8.6. 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

Directory Level	Directory Name	Description

Level 1	500_GTModelGeometry	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 3rd, 4th, and 5th 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 *SHALL* be stored in level 4 of the 5-level directory structure presented above. The names of the files *SHALL* adhere to the following naming convention: D500_Snnn_Tnnn_FeatureCode_FSC_MODL".ext[33]"

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
ext	The file type associated with the dataset (For an OpenFlight file this is ".flt")

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 *SHALL* be stored in level 5 of its directory structure. The names of the files *SHALL* adhere to the following naming convention: D510_Snnn_Tnnn_LOD_FeatureCode_FSC_MODL.<ext[34]>

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
ext	The file type associated with the dataset (For an OpenFlight file this is ".flt")

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 *SHALL* be stored in level 4 of the 5-level directory structure presented above. The names of the files *SHALL* 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).

Assuming the use of OpenFlight, 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 Libray; 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

Directory 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 *SHALL* adhere to the following naming convention: D511_Snnn_Tnnn_LOD_TNAM.<ext>

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.
ext	The file type associated with the dataset (For CDB Version 1.0 this is a SGI Image with file extension rgb.)

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 *SHALL* adhere to the following naming convention: D504_Snnn_Tnnn_LOD_TNAM.<ext>

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.
ext	The file type associated with the dataset (In CDB Version 1 specified as a TIFF file <.tif>)

3.4.2.3. GTModelCMT Naming Convention

Requirement 51	http://www.opengis.net/spec/cdb/core/1.0/gtmodelcmt-name The names of the GTModelCMT files *SHALL* 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 assuming SGI rgb based textures, 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 metadata (if provided) associated with the above material textures would reside in the same directory and be named:

.\D511_S001_T001_Church-Gothic_mtd.xml

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 assuming images as TIFF files, the following files would be found in the above directory:

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

The metadata (if provided) associated with the above material textures would reside in the same directory and be named:

3.4.3. GTModel Directory Structure 3: Interior Geometry and Descriptor

D505_Snnn_Tnnn_Church-Gothic_mtd.xml

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 3rd, 4th, and 5th 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 *SHALL* be stored in level 5 of its directory structure. The names of the files *SHALL* adhere to the following naming convention: D506_Snnn_Tnnn_LOD_FeatureCode_FSC_MODL.”ext”

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
ext	The file type associated with the dataset (For an OpenFlight file the extension is ".flt")

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 *SHALL* be stored in level 4 of the 5-level directory structure presented above. The names of the files *SHALL* adhere to the following naming convention: D508_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 3-12: 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
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.

3.4.3.3. 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\L03\

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 files [35] representing LOD 3 of these buildings.

Assuming the use of OpenFlight, 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

If there is also additional metadata for the interior of these models, they would also 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_mtd.xml

.\D508_S001_T001_AL015_015_School_mtd.xml
.\D508_S001_T001_AL015_021_Garage_mts.xml
.\D508_S001_T001_AL015_037_Fire_Station_mtd.xml
.\D508_S001_T001_AL015_050_Church_mtd.xml

Note that any one of the models may have additional metadata or all may have additional metadata.

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.

Table 3-13: GTModelInteriorTexture Directory Structure

Directory 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 *SHALL* adhere to the following naming convention: D507_Snnn_Tnnn_LOD_TNAM.<ext>

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.
<ext>	The file type associated with the dataset (In CDB version 1.0 this is an SGI Image with a rgb file extension)

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 *SHALL* adhere to the following naming convention: D509_Snnn_Tnnn_LOD_TNAM.<ext>

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.
ext	The file type associated with the dataset (In CDB Version 1.0 specified as a TIFF file - tif)

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 SGI RGB image file format is used, 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

The metadata (if provided) associated with the above material textures would reside in the same directory and be named:

```
.\D509_Snnn_Tnnn_Church-Gothic_mtd.xml
```

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

Directory 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 3rd, 4th, and 5th 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 *SHALL* adhere to the following naming convention: D512_Snnn_Tnnn_LOD_FeatureCode_FSC_MODL.<ext>

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.
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
ext	The file type associated with the dataset. This is designated by the file extension: e.g. .tif for TIFF, .flt for OpenFlight, .shp for ShapeFiles, .gkpg for GeoPackage 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 buildings [36].

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

If there is metadata for this file and the metadata is encoded in XML, the above directory would also include:

3.4.6. GTModel Complete Examples

\D512_S001_T017_L03_AL015_004_Castle_mtd.xml

Assuming the use of ShapeFiles, OpenFlight, rgb texture files, and TIFF 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)

\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

Directory 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 [37].
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 Catagory 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 *SHALL* adhere to the following naming convention: D600_Snnn_Tnnn_MMDC.<ext>

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
ext	The file type associated with the dataset (In Version 1 this was anOpenFlight file with the extension is “.flt”)

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.flt
- 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

Directory 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 *SHALL* adhere to the following naming convention: D601_Snnn_Tnnn_Wnn_TNAM.<ext>

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
<ext>	The file type associated with the dataset (For CDB Version 1.0 these are defined as a SGI Image with .rgb extension)

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 *SHALL* adhere to the following naming convention: D604_Snnn_Tnnn_Wnn_TNAM.<ext>

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
ext	The file type associated with the dataset (In CDB Version 1.0 this is defined as a TIFF File - tif)

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

Directory 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 [38].
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	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.<ext> where <ext> 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
ext	'ext' 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.

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_Snnn_Tnnn_L04_1_1_225_1_1_8_0.shp

D606_Snnn_Tnnn_L04_1_1_225_1_1_8_0.shx

D606_Snnn_Tnnn_L04_1_1_225_1_1_8_0.dbf

D606_Snnn_Tnnn_L04_1_1_225_1_1_8_0.dbt __

3.5.4. MModel Complete Examples

Assuming the use of ShapeFiles, SGI rgb files, and OpenFlight, 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. RMTexture
6. RMDescriptor
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. GSModelCMT
21. GSModelInteriorGeometry
22. GSModelInteriorTexture
23. GSModelInteriorDescriptor
24. GSModelInteriorMaterial
25. GSModelInteriorCMT
26. T2DModelGeometry
27. T2DModelCMT
28. Navigation

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

Table 3-28: Tiled Dataset Directory Structure

Directory 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* ≥ 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* ≥ 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 ≤ *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° 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* ≥ 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* ≥ 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 ≤ *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.

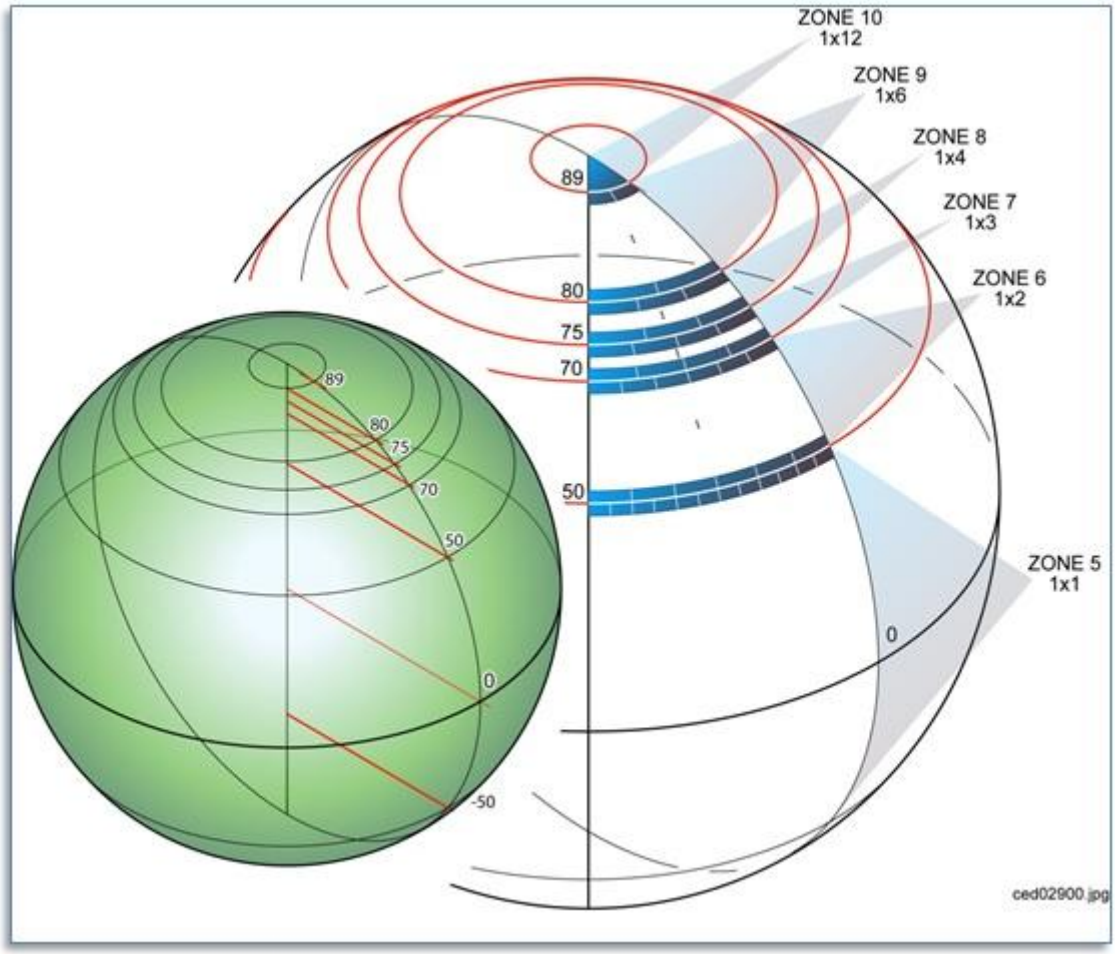

Figure 3-8. Allocation of CDB Geocells with Increasing Latitude

Table 3-29: NbSliceIDIndex for every CDB Zones

Latitude	DLonZone(lat)	NbSliceIDIndex
+89 ≤ lat < +90	12	30
+80 ≤ lat < +89	6	60
+75 ≤ lat < +80	4	90
+70 ≤ lat < +75	3	120
+50 ≤ lat < +70	2	180
–50 ≤ lat < +50	1	360
–70 ≤ lat < –50	2	180
–75 ≤ lat < –70	3	120
–80 ≤ lat < –75	4	90
–89 ≤ lat < –80	6	60
–90 ≤ lat < –89	12	30

3.6.2.2.1. Examples

Data elements located at latitude −5.2° and longitude +45.2° will be found under the directory named:

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

Data elements located at latitude +62.3° 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° 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° 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° 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° 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° 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° 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° 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 $2^{2\times LOD}$. 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:

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

…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*:

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

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

$$DLat = mod(lat + 90, DLatCell)$$

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

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

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

$$DLat = mod(lat + 90,\ 1)$$

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

3.6.2.5.1. Examples

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

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

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

Assuming longitude +45.2°, 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:

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

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

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.<ext>

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.
ext	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:

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

…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*:

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

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

$$DLon = mod(lon + 180, DLonCell)$$

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

$$R_{Ref} = int\left(\frac{DLon \times 2^{Lod}}{DLonCell}\right)$$

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

$$DLon = mod(lon + 180,\ DLonZone(lat))$$

$$R_{Ref} = int\left(\frac{DLon \times 2^{Lod}}{DLonZone(lat)}\right)$$

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° and at LOD 2, the RREF index computed from the above formulas will be:

$$DLon = mod\big(45.2 + 180,\ DLonZone(-5.2)\big) = mod(225.2,\ 1) = 0.2$$

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

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° and longitude -160.4° will produce the following RREF index:

$$DLon = mod\big(-160.4 + 180,\ DLonZone(62.3)\big) = mod(19.6,\ 2) = 1.6$$

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

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. GSModelCMT
6. GSModelInteriorGeometry
7. GSModelInteriorTexture
8. GSModelInteriorMaterial
9. GSModelInteriorDescriptor
10. GSModelInteriorCMT
11. GSModelMetadata

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*".<ext>

The extra tokens are described in the next sections.

3.6.3.2.1. GSModel Geometry File Naming Conventions

Requirement 69	http://www.opengis.net/spec/cdb/1.0/core/gsmodelgeometry-archive-name The files from the GSModelGeometry and GSModelInteriorGeometry datasets *SHALL* have the following naming convention: LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_FeatureCode_FSC_MODL.<ext>[39]

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. GSModel Texture File Naming Conventions

Requirement 70	http://www.opengis.net/spec/cdb/1.0/core/gsmodeltexture-archive-name The files from the GSModelTexture and GSModelInteriorTexture datasets *SHALL* have the following naming convention: LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_TNAM.<ext>[40]

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

3.6.3.2.3. GSModel Material File Naming Conventions

Requirement 71	http://www.opengis.net/spec/cdb/1.0/core/msmodelmaterial-archive-name The files from the GSModelMaterial and GSModelInteriorMaterial datasets *SHALL* have the following naming convention: LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_TNAM.<ext>[41]

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

3.6.3.2.4. GSModel Descriptor File Naming Conventions

Requirement 72	http://www.opengis.net/spec/cdb/1.0/core/gsmodeldescriptor-archive-name The files from the GSModelGeometry and GSModelInteriorGeometry datasets *SHALL* have the following naming convention: LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_FeatureCode_FSC_MODL.<ext>[42]

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. GSModel CMT File Naming Conventions

The files from the GSModelCMT and GSModelInteriorCMT datasets have the following naming convention:

_ LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_TNAM.xml _

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

3.6.3.2.6. 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_D309_S001_T001_L07_U38_R102.zip (CMT)

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)

N62W162_D311_S001_T001_L07_U38_R102.zip (Interior CMT) _

Assuming the use of ShapeFiles, SGI rgb files and OpenFlight, 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, if using OpenFlight 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: 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

Directory Level	**Directory Name**	**Description**
Level 1	400_NavData	The name of the directory is composed of the dataset 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 *SHALL* have the following naming convention: D400_Snnn_Tnnn.<ext>

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
ext	The file type associated with the dataset. (In CDB Version 1.0 this is for a dBASE III+ file or database table - .dbf)

3.7.2.1. Examples

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

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

[23] 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.

[24] As defined in section 2.2, File System

[25] 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).

[26] 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).

[27] The actual equation to obtain the values of 111319 m is L=a×PI180° where “a” is the length of the major semi-axis of the WGS-84 ellipsoid; “a” is also known as the equatorial radius.

[28] Defence Geographic Information Working Group

[29] http://www.sedris.org/

[30] Ultra High Resolution Building

[31] In CDB Version 1, Feature Codes were orginally 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.

[32] 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.

[33] “.ext” represents the file extension used for a given model type. For example, an OpenFlight file extension is “.flt.”

[34] “ext” represents the file extension used for a given model type. For example, an OpenFlight file extension is “.flt.” For CDB Version 1.0, this was always flt.

[35] Currently OpenFlight

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

[37] Volume 2 CDB Core Model and Physical Structure: Informative Annexes

[38] Volume 2 CDB Core: Model and Physical Structure: Informative Annexes

[39] The file extension was .flt in CDB Version 1

[40] The file extension was .rgb in CDB Version 1

[41] The file extension was .tif in CDB Version 1

[42] The file extension was .flt in CDB Version 1

Chapter 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 the following.

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 conformat 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. SGI Image (*.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. The use of this format 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. OpenFlight (*.flt): used for the representation of 3D model geometry. Refer to Volume 6, OGC CDB Rules for Encoding Data using OpenFlight, for details.
6. Shapefile (*.shp/shx/dbf/dbt): used for the representation and attribution of vector data. Refer to Volume 4, OGC CDB Best Practice use of Shapefiles for Vector Data Storage, for details.
7. GeoPackage (*.gpkg): used for the representation and attribution of vector data. Refer to Volume 13, OGC CDB Rules for Structuring a GeoPackage, for details.
8. 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.
9. Cross-platform and interoperable file storage and transfer format (*.zip): used to archive and store geospecific 3D model datasets. The ZIP format is mainly used as a container to regroup files located in a given directory. Compressing ZIP files is allowed. The application creating the file is free to decide whether or not it compresses its content.

For all other formats (used by this standard), CDB readers should be fully compliant [43].

File Format	Minimal Version Number
TIFF	6.0
SGI Image	1.0
JPEG 2000	1.0
OpenFlight	16.0
Shapefile	Esri White Paper, July 98
dBASE	III+
XML	1.0 and later
ZIP	6.3.1 and later
GeoPackage	1.1 and later

Previous version of the above table had a column labeled: *CDB Client-device Behavior for Prior Versions"*. All rows had the label *Ignores data*. The column has been removed from the table but the

value is still valid.

include::clause_10_CDB_Datasets_metadata.adoc
include::clause_10_CDB_Datasets_nav_and_mode_texture.adoc
include::clause_10_CDB_Datasets_tiled_raster_and_imagery.adoc
include::clause_10_CDB_Datasets_Tiled_vector.adoc

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

Annex A: Conformance Class Abstract Test Suite (Normative)

A.1. Conformance Test Class: OGC CDB Core Standard

This section describes conformance test for the OGC CDB Core Standard. 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 1.0 standard.

The conformance class id is “http://www.opengis.net/spec/http://opengis.net/spec/CDB/1.0/core/lod-hierarchy[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	**/conf/core/model**	
Requirements	/req/core/model	
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	**/conf/core/platform**
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 ±90° latitude, and ±180° 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 *is* 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. 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

Geotypical Models LOD Resolution

NOTE	Labelled as A.1.14b Geotypical Models LOD Resolution in the MS Word version of the standard

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.<ext>. Always flt in CDB Version 1,0
	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.<ext>. In CDB Version 1.0 this was always ".flt" for OpenFlight files.
	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 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.<ext> Always rgb in CDB Version 1,0
	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.<ext> Always tif in CDB Version 1,0
	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.<ext>. For CDB Version 1.0 this was always ".flt" for OpenFlight files.
	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_FeatureCode_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.<ext> Always rgb in CDB Version 1,0
	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.<ext> Always tif in CDB Version 1,0
	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.ext
	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.".ext". ".flt" for OpenFlight files.
	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.<ext> Always rgb in CDB Version 1,0
	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.<ext>. Always tif in CDB Version 1,0
	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.ext where ext 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.ext
	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.

[cols=",,",

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".<ext>
	Test method	Visual
	Test type	Conformance
Test 69	/conf/core/archive/gsmodelgeometry-archive-name	
	Requirement	/req/core/gsmodelgeometry-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_FeatureCode_FSC_MODL.<ext> “.flt” for OpenFlight files.
	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.<ext> For CDB Version 1.0 this is 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.<ext> For CDB Version 1.0 this is 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_FeatureCode_FSC_MODL.<ext>For Version 1.0 this is ".flt" for OpenFlight files.
	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 NacData 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.

[cols=",,",

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.

[cols=",,",

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.

[cols=",,",

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 *Mesh Type* channel of the TIFF image contains the necessary components and *are* stored in correct format.
	Test method	Visual
	Test type	Conformance
Test 130	/conf/core/tiled-raster-elevation-terrain/terrain-tiff-channel3	
	Requirement	/req/core/terrain-tiff-channel3
	Test purpose	Verify that the *Latitude Offset* and *Longitude Offset* channels of the TIFF image contains the necessary components and *are* stored in correct format.
	Test method	Visual
	Test type	Conformance
Test 91	/conf/core/tiled-raster-elevation-terrain/terrain-hyspography-offline	
	Requirement	/req/core/terrain-hyspography-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 *is* a type PointZ, a MultiPointZ, a PolyLineZ, a PolygonZ or a `MultiPatch (see note)`. `Note: Geometry type multi-patch was deprecated in version 1.2 of this standard. This geometry is no longer supported. Use at your own risk. This geometry type will remain in the CDB standard until version 2.0.`
	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 *is* 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 *are* 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.

[cols=",",

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 (Deprecated in version 1.1. No longer a requirement	/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 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 *is* 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 *is* 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 (see note) feature, the AHGT attribute is set to TRUE. Vector feature types implemented as Point, PointM, MultiPoint, MultiPointM, PolyLine, PolyLineM, Polygon and PolygonM are ignored. Note: Geometry type multi-patch was deprecated in version 1.2 of this standard. This geometry is no longer supported. Use at your own risk. This geometry type will remain in the CDB standard until version 2.0.
	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/hyspograph-vector-types	
	Requirement	/req/core/hyspograph-vector-types

	Test purpose	Verify that vector features are of type PointZ, MultiPointZ, PolyLineZ, PolygonZ, or `MultiPatch (see note)`. `Note: Geometry type multi-patch was deprecated in version 1.2 of this standard. This geometry is no longer supported. Use at your own risk. This geometry type will remain in the CDB standard until version 2.0.`
	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 one 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 one VCMT for each CDB Geocell.
	Test type	Conformance

A.1.37. GeoSpecific Model Descriptor

The following conformance class is designed to determine if the CDB geocells have one GeoSpecific Model Descriptor placed at the correct LOD for each GeoSpecific Model's level of detail.

Conformance Class	**/conf/core/gsmd**	
Requirements	/req/core/gsmd	
Dependency	Various XML schema	
Test 129	**Test purpose**	Verify that there is one GS Model Descriptor file for each GS Model level of detail, and that the file is found at the same LOD as the model's level of detail.
	Test method	Visually pass if the GS Model Descriptor file exists at the proper LOD for each GS Model's level of detail file.
	Test type	Conformance

Annex B: Revision History

Date	Release	Editor	Primary clauses modified	Description
2016-04-04	1.0	C. Reed	all	Initial version
2018-03-04	1.1	C. Reed	all	revision

Annex C: Bibliography

This table lists the documentation referenced throughout this document.

Ref	Title	Description
1	ARINC Standard 424-16	Navigation System Data Base, Aeronautical Radio Inc., August 30, 2002.
2	ASTARS-04 CDB	Systems Requirements
3	Digital Geographic Information Exchange Standard (DIGEST), Standard V2.1	The document can be obtained at the following address: **http://www.digest.org/**
4	Enumeration and Bit Encoded Values for use with Protocols for Distributed Interactive Simulation Applications.	This is document SISO-REF-010. It accompanies IEEE Std 1278.1-1995 and can be obtained from the *Simulation Interoperability Standards Organization* at **http://www.sisostds.org/**
5	Extensible Markup Language (XML) 1.0 (Third Edition)	Bray, Tim, et al. **http://www.w3.org/TR/2004/REC-xml-20040204/** W3C Recommendation, February 04, 2004.
6	Guide - PD6777, BSI's _Guide to the practical implementation of JPEG 2000 _	The document can be found at: **http://www.jpeg.org/** Other useful sites include: **http://en.wikipedia.org/wiki/SRGB_color_space** The document is targeted at managers; application software developers and end-users who want to know more about JPEG 2000.
7	IEEE Standard for Distributed Interactive Simulation - Application Protocols	IEEE Std 1278.1-1995

8	JPEG 2000: Image Compression Fundamentals, Standards and Practice	Kluwer International Series in Engineering and Computer Science, Secs 642, by David S. Taubman and Michael W. Marcellin
9	MIL-STD-2411 Raster Product Format Specification	The Raster Product Format (RPF) is a standard data structure for geospatial databases composed of rectangular arrays of pixel values (e.g., in digitized maps or images) in compressed or uncompressed form. RPF defines a common format for the interchange of raster-formatted digital geospatial data among DoD Components. Department of Defense Information Technology Standards Registry Baseline Release 04-2.0. http://earth-info.nga.mil/publications/specs/printed/2411/2411_RPF.pdf
10	MIL-C-89041 Controlled Image Base Specification	Controlled Image Base (CIB). This Specification provides requirements for the preparation and use of the RPF CIB data. CIB is a dataset of orthophotos, made from rectified grayscale aerial images. http://www.fas.org/irp/program/core/mil-c-89041-cib.htm
11	OpenFlight Scene Description Database Standard, Version 16.0, Revision A, November 2004, Presagis Inc	The original document has been annotated by CAE to create the CDB-annotated OpenFlight Standard.
12	Product Standard for the Digital Aeronautical Flight Information File (DAFIF), Eight Edition, Doc. # PS/1FD/086	National Imagery and Mapping Agency (NIMA), April 2003.

13	SEDRIS™ - Synthetic Environment Data Representation Interchange Specification	The Source for Synthetic environment Representation and Interchange. http://www.sedris.org
14	Shapefile Technical Description - an ESRI White Paper—July 1998	The original document has been annotated by CAE Inc to create the CDB-annotated Shapefile Technical Description.
15	The SGI Image File Format, Version 1.00, Paul Haeberli, Silicon Graphics Computer Systems	This specification can be found at: http://paulbourke.net/dataformats/sgirgb/sgiversion.html
16	TIFF rev 6.0 Adobe Developers Association, Adobe Systems Incorporated, 1585 Charleston Road and P.O. Box 7900Mountain View, CA 94039-7900	A copy of this original standard can be found at: http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf and at: ftp://ftp.adobe.com/pub/adobe/DeveloperSupport/TechNotes/PDFfiles The original document has been annotated by CAE Inc to create the CDB-annotated TIFF Standard.
17	XML Schema Part 0: Primer Second Edition	Fallside, David, Priscilla Walmsley. **http://www.w3.org/TR/xmlschema-0/** W3C Recommendation, October 28, 2004.

18	XML Schema Part 1: Structures Second Edition	Thompson, Henry S., et al. **http://www.w3.org/TR/xmlschema-1/** W3C Recommendation, October 28, 2004.
19	XML Schema Part 2: Datatypes Second Edition	Biron, Paul V., Ashok Malhotra. **http://www.w3.org/TR/xmlschema-2/** W3C Recommendation, October 28, 2004.
20	ICAO Airline Designator	List of ICAO Airline Codes, **http://en.wikipedia.org/wiki/Airline_codes**
21	Radar Signatures and Relations to Radar Cross-Section. Mr. P E R Galloway, Roke Manor Research Ltd, Romsey, Hampshire, United Kingdom.	This document can be obtained at the following Internet address: **http://aircraftdesign.nuaa.edu.cn/lo/Ref/General%20Topics/radar_signatures_and_relations_to_rcs.pdf**
22	AN/APA to AN/APD - Equipment Listing.	This document can be obtained at the following Internet address: **http://www.designation-systems.net/usmilav/jetds/an-apa2apd.html#_APA**
23	Radar Polarimetry - Fundamentals of Remote Sensing. National Resources Canada.	This document can be obtained at the following Internet address: **https://www.nrcan.gc.ca/earth-sciences/geomatics/satellite-imagery-air-photos/satellite-imagery-products/educational-resources/9275**

24	RCS in Radar Range Calculations for Maritime Targets, by Ingo Harre. Bremen, Germany. (V2.0-20040206).	This document can be obtained at the following Internet address: **http://www.mar-it.de/Radar/RCS/RCS_xx.pdf**
25	Decibels relative to a square meter – dBsm. By Zhao Shengyun.	This document can be obtained at the following Internet address: **http://radarproblems.com/chapters/ch06.dir/ch06pr.dir/c06p11.dir/c06p11.htm**
26	MIL-C-89041	Controlled Image Base (CIB)
27	MIL-STD-2411	Defense Mapping Agency, Military Standard, Raster Product Format (RPF)
28	MIL-STD-2411-1	Defense Mapping Agency, Registered Data Values for Raster Product Format
29	MIL-STD-2411-2	Defense Mapping Agency, Incorporation of Raster Product Format (RPF) Data in National Imagery Transmission Format (NITF).
30	IEEE Std 1516-2000	IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)
31	RPR-FOM Version 2 Draft 17	Real-time Platform Reference (RPR) Federation Object Model (FOM) This RPR-FOM maps the DIS standard to the HLA standard. The document can be obtained from the *Simulation Interoperability Standards Organization* at the following address: **http://www.sisostds.org/**

32	MIL-PRF-89039 Amendment 2	Performance Specification Vector Smart Map (VMAP Level 0), 28 September 1999 **http://en.wikipedia.org/wiki/Vector_Map http://earth-info.nga.mil/publications/specs/printed/VMAP0/vmap0.html**
33	MIL-PRF-89033 Amendment 1	Performance Specification Vector Smart Map (VMAP Level 1), 27 May 1998
34	MIL-PRF-89035A	Urban Vector Map (UVMap), 1st August, 2002
35	OneSAF Ultra High Resolution Building (UHRB) Object Model	OneSAF UHRB Object Model (Version 2.2, Document Revision E, March 7th, 2008, Contract #: N61339-00-D-0710, Task Order: 28.) *http://www.onesaf.net/community/systemdocuments/v.3.0/MaintenanceManual/erc/UHRB_2_Object_Model.pdf *
36	OneSAF Ultra High Resolution Building (UHRB) On-Disk Format	OneSAF UHRB On-Disk Format Model (Version 2.2, Document Revision E, March 7th, 2008, Contract #: N61339-00-D-0710, Task Order: 28.) *http://www.onesaf.net/community/systemdocuments/v.3.0/MaintenanceManual/erc/UHRB_2_On_Disk_Format.pdf *
37	U.S. Department of Transportation - Federal Aviation Administration – Advisory Circular 150/5340-1J	Standards for Airport Markings, AC- 150/5340-1J, dated 4/29/2005
38	Federal Aviation Administration – Aeronautical Information Manual	Official Guide to Basic Flight Information and ATC Procedures, dated 14th February, 2008