

# COMMON DATABASE SPECIFICATION

Volume 1

Version 3.2

Update 1

22 February 2016



**PRESAGIS**

## Copyright

### Common Database (CDB)

© 2016 Presagis. All Rights Reserved.

THIS DOCUMENT AND ITS CONTENT (“INFORMATION”) ARE PROVIDED "AS IS" WITHOUT WARRANTY OR CONDITION OF ANY KIND. USE OF THE INFORMATION IS AT YOUR OWN RISK. PRESAGIS DOES NOT MAKE ANY REPRESENTATION OR WARRANTY ABOUT THE QUALITY, ACCURACY, RELIABILITY, COMPLETENESS OR CURRENCY OF THE INFORMATION. PRESAGIS DOES NOT ASSUME ANY RESPONSIBILITY FOR ANY ERROR, OMISSION OR INACCURACY IN THE INFORMATION. IN NO EVENT SHALL PRESAGIS BE LIABLE FOR ANY DAMAGE RESULTING FROM RELIANCE ON OR USE OF THE INFORMATION.

You may, free of charge, further distribute the Information or any portion thereof without any restriction, on the conditions that You:

- make no modification to the Information without Presagis’ prior written consent,
- keep intact all proprietary notices, and
- provide attribution to Presagis when the Information is used for publication purposes.

Unless in the public domain or specifically credited to another copyright holder, Presagis is the owner of all intellectual property rights in and to the Information. All trademarks contained in this document are the property of their respective owners.

## Revision History

The new revision history is presented in reversed chronological order where the most recent revision appears at the top of the table.

Version	Date	Description
3.2 Update 1	22 February 2016	<p>Changes:</p> <ul style="list-style-type: none"> <li>- Corrected the definition of GSModelCMT</li> <li>- Added GSModelInteriorCMT</li> <li>- Added new contributors to Chapter 11</li> <li>- Added a note to Chapter 4 to allow compressing ZIP files</li> <li>- Added Section 5.8.1.1 to define a limit on GSModel archives</li> <li>- Added Dataset 312, T2DModelCMT</li> <li>- Corrected the Feature Data Dictionary</li> <li>- Simplified Section 6.8</li> <li>- Added Section 5.7.1.6.4 and 5.7.1.9</li> <li>- Added Appendix A.21</li> <li>- Added a Priority to FACC codes (in the FDD)</li> <li>- Added countries to Appendix I</li> <li>- Updated CDB_Attributes.xml</li> <li>- Restored the NVT Attribute</li> <li>- Updated the WGP Attribute</li> <li>- Changed Restriction 3 of Section 6.2.2.1</li> <li>- Addition of new Airliners to Appendix O</li> <li>- Added new base materials for use in building interiors</li> <li>- Added the 'update' attribute to Version.xml</li> </ul>
3.2	19 March 2014	<p>Changes:</p> <ul style="list-style-type: none"> <li>- Updated the list of DIS codes found in /CDB/Metadata/Moving_Model_Codes.xml</li> <li>- Editorial changes to volume 1 and 2 to remove markups and highlights, promote a consistent use of various terms and expressions, check spelling, correct the formatting, etc.</li> </ul>
	12 February 2014	<p>First public release of version 3.2 of the CDB Specification.</p> <p>The changes with respect to the first release candidate of 19 December 2013 are the followings:</p> <ul style="list-style-type: none"> <li>- Revised the implementation of the Primary Alternate Elevation (formerly Subordinate Terrain Offset)</li> <li>- Added the Subordinate Alternate Bathymetry</li> <li>- Introduced the concept of Mesh Type and added an optional channel to the Primary Terrain Elevation and Subordinate Bathymetry components to store the Mesh Type.</li> </ul>



	19 December 2013	First release candidate of version 3.2 of the CDB Specification.  - The changes with respect to the first draft of December 2012 take into account comments received from users and address concerns about compatibility with version 3.0.
	December 2012	First Draft of version 3.2 of the CDB Specification.

### Old Revision History

The old revision history is presented in chronological order where the most recent revision appears at the bottom of the table.

Revision Level	Date	Description
V1.0 – First Draft	October 28, 2005	First draft available for comments from industry
V2.0 – Second Draft	March 16, 2006	Second draft of the CDB Specification
V2.1 – Third Draft	October 25, 2006	Third draft of the CDB Specification
V2.2 – Fourth Draft	June 15, 2007	Fourth draft of the CDB Specification
V2.3 – Fifth Draft	July 9, 2007	Fifth draft of the CDB Specification
V2.4 – Sixth Draft	October 5, 2007	Sixth draft of the CDB Specification
V2.5 – Seventh Draft	November 9, 2007	Seventh draft of the CDB Specification
V3.0 – Draft	December 21, 2007	First draft of CDB 3.0
V3.0 – Draft 2	March 27, 2008	Second draft of CDB 3.0
V3.0 – Draft 3	June 25, 2008	Third draft of CDB 3.0
V3.0	September 2008	Official release of CDB 3.0
V3.1 – Draft	November 2009	First draft of CDB 3.1
V3.1	May 2010	Official Release of CDB 3.1
V3.1 – Correction 1	December 2011	Editorial Corrections to CDB 3.1



## Abstract

The CDB Specification provides the means for a single, versionable, simulation-rich, synthetic representation of the earth. A database that conforms to this Specification is referred to as a **C**ommon **D**ata**B**ase (CDB). A CDB provides for a synthetic environment repository that is plug-and-play interoperable between database authoring workstations. Moreover, a CDB can be used as a common on-line (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 at runtime.

The application of CDB to future simulator architectures will significantly reduce runtime-source level and algorithmic correlation errors, while reducing development, update and configuration management timelines. With the addition of the HLA/FOM and DIS protocols, the application of the CDB Specification provides a Common Environment to which inter-connected simulators share a common view of the simulated environment.

The CDB Specification is an open format Specification for the storage, access and modification of a synthetic environment database. The Specification defines the data representation, 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 Specification makes use of several commercial and simulation data formats endorsed by leaders of the database tools industry.

The CDB synthetic environment is a representation of the natural environment including external features such as man-made structures and systems. It encompasses the 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 synthetic environment includes the specific attributes of the synthetic environment data as well as their relationships.

A CDB 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 is specifically tailored for real-time applications.

## **Distribution Package**

The CDB Specification is distributed as an archive whose name is:

CDB Specification – Version x.y[.z].zip

The archive contains two root folders:

/Documents/

contains the PDF documents making up the Specification

/CDB/

illustrates the directory structure and filenaming conventions of a CDB; it also contains the `Metadata` folder where developers will find important XML files

The important files of the distribution package are the followings:

/Documents/CDB Specification – Volume 1.pdf

/Documents/CDB Specification – Volume 2.pdf

/CDB/Metadata/\*.xml

## Table of Contents

<b>1</b>	<b>Introduction.....</b>	<b>1-1</b>
1.1	Purpose.....	1-1
1.2	Document Structure .....	1-1
1.3	Scope.....	1-2
1.3.1	What is the CDB Specification .....	1-3
1.3.1.1	Use of CDB as an Off-line Database Repository.....	1-5
1.3.1.2	Use of CDB as a Combined Off-line and Run-time Database Repository .....	1-9
1.3.2	What the CDB Specification Is Not.....	1-12
1.3.3	What is a CDB .....	1-13
1.4	Key Features and Characteristics of the CDB Specification .....	1-14
1.4.1	Synthetic environment Database for Simulation Applications .....	1-14
1.4.2	Logical Addressability .....	1-14
1.4.3	High Spatial Resolution and Scalability .....	1-15
1.4.4	Earth Geodetic Spatial Representation Model.....	1-15
1.4.5	Tile/Layer/Level-of-Detail Structure .....	1-15
1.4.5.1	Tiles.....	1-15
1.4.5.2	Layers.....	1-16
1.4.5.3	Levels-of-Detail .....	1-16
1.4.6	Platform Independence .....	1-17
1.4.6.1	System Software Independence .....	1-18
1.4.6.2	System Hardware Independence.....	1-19
1.4.7	Synthetic Environment Scalability & Adaptability .....	1-22
1.4.8	Platform Scalability .....	1-24
1.4.9	Simulator Wide Unique Data Representation, Data Normalization .....	1-26
1.4.10	Compression of Storage Intensive Imagery Datasets .....	1-27
1.4.11	Compression of other Raster Datasets .....	1-28
1.5	Key Benefits of the CDB Specification.....	1-28
1.5.1	Improved Synthetic environment DB Generation Timeline .....	1-28
1.5.2	Interoperable Simulation-Ready Synthetic environment DB .....	1-29
1.5.3	Improved Client-device Robustness/Determinism .....	1-30
1.5.4	Runtime-Adjustable Synthetic Environment DB Correlation and Fidelity ...	1-30
1.5.5	Increased Synthetic Environment DB Longevity .....	1-31
1.5.6	Reduced Synthetic Environment DB Storage Infrastructure Cost.....	1-31
1.6	CDB Primer .....	1-32
1.6.1	CDB Specification Data Representation and Organization .....	1-32
1.6.2	CDB Specification Logical Structure .....	1-34
1.6.3	CDB Structure, Organization on Media and Conventions.....	1-35
1.6.4	Typical Implementation on a Simulator .....	1-35
1.6.4.1	Database Generation Facility .....	1-37
1.6.4.2	Database Generation Flow .....	1-37
1.6.4.3	Update Manager.....	1-41
1.6.4.4	CDB Servers .....	1-42
1.6.4.5	Other Applications of the CDB Specification .....	1-46



1.6.5	Use of CDB in Applications Requiring Dynamic Synthetic Environments..	1-47
1.6.6	Synthetic Environment Database Correlation.....	1-48
<b>2</b>	<b>CDB Concepts .....</b>	<b>2-1</b>
2.1	Partitioning the Earth into Tiles.....	2-1
2.1.1	Description.....	2-2
2.1.2	Tile Levels-of-Detail (Tile-LODs) .....	2-5
2.1.2.1	Tile-LOD Area Coverage Rules .....	2-9
2.1.2.2	Tile-LOD Hierarchy Rules .....	2-10
2.1.2.3	Tile-LOD Replacement Rules.....	2-10
2.1.3	Handling of the North and South Pole.....	2-11
2.2	File System Requirements .....	2-11
2.2.1	Character Set.....	2-12
2.2.2	A word about case-sensitiveness .....	2-13
2.3	Light Naming.....	2-13
2.3.1	Adding Names to the CDB Light Name Hierarchy .....	2-16
2.4	Model Component Naming.....	2-16
2.4.1	Adding New Model Components .....	2-17
2.5	Materials .....	2-17
2.5.1	Base Materials.....	2-18
2.5.1.1	Base Material Table (BMT).....	2-19
2.5.2	Composite Materials .....	2-19
2.5.2.1	Composite Material Substrates .....	2-19
2.5.2.2	Composite Material Tables (CMT).....	2-21
2.5.2.3	Example 1 .....	2-23
2.5.2.4	Example 2 .....	2-23
2.5.3	Bringing it all Together.....	2-23
2.5.4	Determination of Material Properties by SEM .....	2-24
2.5.4.1	Example .....	2-26
2.5.5	Generation of Materials for Inclusion in CDB Datasets.....	2-26
<b>3</b>	<b>CDB Structure.....</b>	<b>3-1</b>
3.1	Top-Level CDB Structure Description .....	3-1
3.1.1	Metadata Directory .....	3-2
3.1.2	Metadata File Examples.....	3-4
3.2	CDB Configuration Management .....	3-4
3.2.1	CDB Version.....	3-4
3.2.1.1	CDB Extensions.....	3-6
3.2.2	CDB Version Directory Structure.....	3-6
3.2.3	CDB File Replacement Mechanism.....	3-7
3.2.3.1	How to Handle Archives.....	3-9
3.2.3.2	How to Handle the Metadata Directory .....	3-9
3.2.4	CDB Configuration.....	3-9
3.2.5	Management of CDB Configurations and Versions .....	3-10
3.3	CDB Model Types .....	3-10
3.3.1	GTModel (Geotypical 3D Model) .....	3-11

3.3.2	GModel (Geospecific 3D Model).....	3-11
3.3.3	T2DModel (Tiled 2D Model) .....	3-12
3.3.4	MModel (Moving 3D Model).....	3-12
3.3.5	Use of GModels and GTModels.....	3-12
3.3.6	Organizing Models into Levels of Details .....	3-16
3.3.7	Organizing Models into Datasets.....	3-17
3.3.8	Terms and Expressions .....	3-18
3.3.8.1	Feature Classification.....	3-18
3.3.8.2	Model Name.....	3-18
3.3.8.3	DIS Entity Type .....	3-19
3.3.8.4	Texture Name.....	3-19
3.3.8.5	Level of Detail .....	3-20
3.4	GModel Library Datasets .....	3-20
3.4.1	GModel Directory Structure 1: Geometry and Descriptor .....	3-20
3.4.1.1	GModelGeometry Entry File Naming Convention .....	3-21
3.4.1.2	GModelGeometry Level of Detail Naming Convention .....	3-22
3.4.1.3	GModelDescriptor Naming Convention .....	3-23
3.4.1.4	Examples.....	3-23
3.4.2	GModel Directory Structure 2: Texture, Material, and CMT .....	3-24
3.4.2.1	GModelTexture Naming Convention.....	3-25
3.4.2.2	GModelMaterial Naming Convention.....	3-25
3.4.2.3	GModelCMT Naming Convention.....	3-26
3.4.2.4	Examples.....	3-26
3.4.3	GModel Directory Structure 3: Interior Geometry and Descriptor .....	3-27
3.4.3.1	GModelInteriorGeometry Naming Convention .....	3-28
3.4.3.2	GModelInteriorDescriptor Naming Convention .....	3-29
3.4.3.3	Examples.....	3-29
3.4.4	GModel Directory Structure 4: Interior Texture, Material, and CMT .....	3-30
3.4.4.1	GModelInteriorTexture Naming Convention.....	3-31
3.4.4.2	GModelInteriorMaterial Naming Convention.....	3-31
3.4.4.3	Example 1 .....	3-32
3.4.4.4	Example 2 .....	3-33
3.4.5	GModel Directory Structure 5: Signature .....	3-33
3.4.5.1	Naming Convention .....	3-34
3.4.5.2	Examples.....	3-35
3.4.6	GModel Complete Examples .....	3-35
3.5	MModel Library Datasets .....	3-36
3.5.1	MModel Directory Structure 1: Geometry and Descriptor.....	3-36
3.5.1.1	MModelGeometry Naming Convention .....	3-37
3.5.1.2	MModelDescriptor Naming Convention .....	3-38
3.5.1.3	Examples.....	3-38
3.5.2	MModel Directory Structure 2: Texture, Material, and CMT .....	3-38
3.5.2.1	MModelTexture Naming Convention.....	3-39
3.5.2.2	MModelMaterial Naming Convention .....	3-39
3.5.2.3	MModelCMT Naming Convention .....	3-40



3.5.2.4	Examples.....	3-40
3.5.3	MModel Directory Structure 3: Signature .....	3-41
3.5.3.1	Naming Convention .....	3-42
3.5.3.2	Examples.....	3-42
3.5.4	MModel Complete Examples .....	3-43
3.6	CDB Tiled Datasets .....	3-43
3.6.1	Tiled Dataset Types .....	3-43
3.6.1.1	Raster Datasets.....	3-44
3.6.1.2	Vector Datasets .....	3-44
3.6.1.3	Model Datasets.....	3-45
3.6.2	Tiled Dataset Directory Structure .....	3-46
3.6.2.1	Directory Level 1 (Latitude Directory).....	3-47
3.6.2.2	Directory Level 2 (Longitude Directory).....	3-48
3.6.2.3	Directory Level 3 (Dataset Directory) .....	3-51
3.6.2.4	Directory Level 4 (LOD Directory).....	3-52
3.6.2.5	Directory Level 5 (UREF Directory).....	3-52
3.6.3	Tiled Dataset File Naming Conventions.....	3-53
3.6.3.1	File Naming Convention for Files in Leaf Directories (UREF Directory)....	3-53
3.6.3.2	File Naming Convention for Files in ZIP Archives.....	3-55
3.7	Navigation Library Dataset.....	3-58
3.7.1	NavData Structure.....	3-58
3.7.2	Naming Convention .....	3-58
3.7.2.1	Examples.....	3-58
<b>4</b>	<b>CDB File Formats .....</b>	<b>4-1</b>
<b>5</b>	<b>CDB Datasets.....</b>	<b>5-1</b>
5.1	Metadata Datasets .....	5-1
5.1.1	Light Name Hierarchy Metadata .....	5-2
5.1.1.1	Client Specific Lights Definition Metadata .....	5-3
5.1.2	Model Components Definition Metadata.....	5-6
5.1.3	Base Material Table .....	5-7
5.1.4	Default Values Definition Metadata .....	5-7
5.1.5	Specification Version Metadata – Deprecated.....	5-8
5.1.6	Version Metadata .....	5-8
5.1.7	CDB Attributes Metadata .....	5-9
5.1.7.1	Definition of the <Attribute> Element.....	5-9
5.1.7.2	Definition of the <Unit> Element .....	5-10
5.1.7.3	Definition of the <Scaler> Element .....	5-11
5.1.7.4	Example of CDB_Attributes.xml.....	5-12
5.1.8	Geomatics Attributes Metadata.....	5-12
5.1.9	Vendor Attributes Metadata.....	5-12
5.1.10	Configuration Metadata .....	5-13
5.1.10.1	A Note about Folder Path .....	5-13
5.1.10.2	Example .....	5-13
5.2	Navigation Library Datasets .....	5-14

5.2.1	Schema Files .....	5-19
5.2.1.1	Example .....	5-19
5.2.2	Key Datasets .....	5-21
5.2.2.1	Example .....	5-21
5.3	CDB Model Textures .....	5-22
5.4	GTModel Library Datasets .....	5-24
5.5	MModel Library Datasets .....	5-26
5.6	Tiled Raster Datasets .....	5-28
5.6.1	Tiled Elevation Dataset.....	5-33
5.6.1.1	Terrain Mesh Types .....	5-34
5.6.1.2	List of all Elevation Dataset Components .....	5-35
5.6.1.3	Primary Terrain Elevation Component .....	5-37
5.6.1.4	Primary Alternate Terrain Elevation Component .....	5-39
5.6.1.5	Terrain Constraints.....	5-41
5.6.1.6	MinElevation and MaxElevation Components .....	5-45
5.6.1.7	MaxCulture Component.....	5-54
5.6.1.8	Subordinate Bathymetry Component.....	5-56
5.6.1.9	Subordinate Alternate Bathymetry Component.....	5-59
5.6.1.10	Subordinate Tide Component .....	5-60
5.6.2	Tiled Imagery Dataset.....	5-64
5.6.2.1	Raster-Based Imagery File Storage Extension Naming.....	5-64
5.6.2.2	List of all Imagery Dataset Components .....	5-71
5.6.2.3	Visible Spectrum Terrain Imagery (VSTI) Components.....	5-72
5.6.2.4	Visible Spectrum Terrain Light Map (VSTLM) Component .....	5-75
5.6.3	Tiled Raster Material Dataset .....	5-76
5.6.3.1	List of all Raster Material Dataset Components .....	5-79
5.6.3.2	Composite Material Index Component .....	5-79
5.6.3.3	Composite Material Mixture Component .....	5-80
5.6.3.4	Composite Material Table Component.....	5-81
5.7	Tiled Vector Datasets.....	5-82
5.7.1	Introduction to Vector Datasets .....	5-82
5.7.1.1	Shapefile Type Usage and Conventions .....	5-85
5.7.1.2	CDB Attribution.....	5-87
5.7.1.3	CDB Attributes .....	5-95
5.7.1.4	Explicitly Modeled Representations .....	5-138
5.7.1.5	Implicitly Modeled Representations .....	5-139
5.7.1.6	Handling of Topological Networks .....	5-139
5.7.1.7	Handling of Light Points.....	5-143
5.7.1.8	Allocation of CDB Attributes To Vector Datasets .....	5-143
5.7.2	Tiled Navigation Dataset .....	5-147
5.7.2.1	Default Read Value.....	5-148
5.7.2.2	Default Write Value.....	5-148
5.7.3	Tiled GSFeature Dataset .....	5-148
5.7.3.1	Default Read Value.....	5-149
5.7.3.2	Default Write Value.....	5-149





5.7.4	Tiled GTFeature Dataset.....	5-150
5.7.4.1	Default Read Value.....	5-151
5.7.4.2	Default Write Value.....	5-151
5.7.5	Tiled GeoPolitical Feature Dataset.....	5-151
5.7.5.1	Boundary and Location Features.....	5-152
5.7.5.2	Elevation Constraint Features.....	5-153
5.7.5.3	Default Read Value.....	5-155
5.7.5.4	Default Write Value.....	5-155
5.7.6	Tiled RoadNetwork Dataset.....	5-155
5.7.6.1	Default Read Value.....	5-156
5.7.6.2	Default Write Value.....	5-156
5.7.7	Tiled RailRoadNetwork Dataset.....	5-156
5.7.7.1	Default Read Value.....	5-157
5.7.7.2	Default Write Value.....	5-157
5.7.8	Tiled PowerLineNetwork Dataset.....	5-157
5.7.8.1	Default Read Value.....	5-158
5.7.8.2	Default Write Value.....	5-158
5.7.9	Tiled HydrographyNetwork Dataset.....	5-158
5.7.9.1	Default Read Value.....	5-159
5.7.9.2	Default Write Value.....	5-159
5.7.10	Tiled Vector Composite Material Table (VCMT).....	5-160
5.7.10.1	Data Type.....	5-160
5.7.10.2	Default Read Value.....	5-160
5.7.10.3	Default Write Value.....	5-160
5.8	Tiled Model Datasets.....	5-160
5.8.1	Tiled GSModel Datasets.....	5-160
5.8.2	Tiled T2DModel Datasets.....	5-162
<b>6</b>	<b>CDB OpenFlight Models.....</b>	<b>6-1</b>
6.1	OpenFlight File Header.....	6-1
6.2	OpenFlight Model Tree Structure.....	6-1
6.2.1	CDB Model Tree Structure.....	6-3
6.2.2	T2DModel Tree Structure.....	6-3
6.2.2.1	Restrictions.....	6-5
6.2.2.2	Node Attributes.....	6-5
6.2.3	The Use of Node Names.....	6-5
6.2.4	Model Master File.....	6-6
6.2.5	Referencing Other OpenFlight Files.....	6-7
6.2.5.1	Models Straddling Multiple Files.....	6-7
6.2.5.2	Models with Multiple Model-LODs.....	6-8
6.3	Modeling Conventions.....	6-8
6.3.1	Model Coordinate Systems.....	6-8
6.3.1.1	Origin.....	6-9
6.3.1.2	Local Coordinate Systems.....	6-13
6.3.1.3	Units.....	6-13
6.3.1.4	Roll, Pitch, Yaw.....	6-14



6.3.2	Geometry.....	6-14
6.3.3	Roof Tagging .....	6-15
6.3.4	Relative Priority .....	6-15
6.4	Model Identifiers.....	6-16
6.4.1	GSModel and GTModel Identifier.....	6-16
6.4.2	MModel Identifier.....	6-16
6.4.3	2DModel Identifier .....	6-17
6.5	Model Zones .....	6-17
6.5.1	Definition .....	6-17
6.5.2	Global Zones .....	6-18
6.5.3	Zone Attributes .....	6-19
6.5.3.1	Material .....	6-19
6.5.3.2	Temperature .....	6-19
6.5.4	Implementation Guidelines .....	6-20
6.5.5	Model Zone Naming .....	6-23
6.5.6	Usages .....	6-24
6.5.6.1	Model Landing Zones .....	6-24
6.5.6.2	Model Footprint Zones .....	6-24
6.5.6.3	Model Cutout Zones .....	6-26
6.5.6.4	Model Interior Zones .....	6-27
6.6	Model Points .....	6-39
6.6.1	Definition .....	6-39
6.6.2	Usages .....	6-39
6.6.2.1	Model DIS Origin .....	6-39
6.6.2.2	Model Viewpoint .....	6-42
6.6.2.3	Model Attach Point .....	6-43
6.6.2.4	Model Anchor Point.....	6-43
6.6.2.5	Model Center of Mass.....	6-43
6.7	Model Conforming.....	6-44
6.7.1	Non Conformal (Absolute) Mode .....	6-45
6.7.2	Point Conformal Mode .....	6-45
6.7.3	Vertex Conformal Mode.....	6-46
6.7.4	Line Conformal Mode.....	6-47
6.7.5	Plane Conformal Mode .....	6-48
6.7.6	Surface Conformal Mode.....	6-50
6.8	Model Levels-of-Detail.....	6-51
6.8.1	LOD Node Types.....	6-53
6.8.1.1	Note on Additive LODs .....	6-53
6.8.2	LOD Node Ordering .....	6-54
6.8.3	LOD Significant Size.....	6-55
6.8.3.1	Definition of Significant Size .....	6-56
6.8.3.2	Estimating the Size of the Model.....	6-56
6.8.3.3	How to use the Significant Size .....	6-56
6.8.4	LOD Limits.....	6-56
6.8.4.1	How to Assign CDB LODs.....	6-57

6.8.5	LOD Generation Guidelines .....	6-58
6.9	Model Switch Nodes .....	6-58
6.9.1	Definition .....	6-59
6.9.2	Usage .....	6-59
6.9.2.1	Articulations with Discreet Positions .....	6-59
6.9.2.2	Damage States .....	6-59
6.9.2.3	Temporal Anti-aliasing .....	6-62
6.10	Model Articulations .....	6-64
6.10.1	Definition .....	6-64
6.10.2	Usage .....	6-65
6.10.2.1	Rotating Parts .....	6-65
6.11	Model Light Points .....	6-66
6.12	Model Attributes .....	6-67
6.12.1	Definition .....	6-67
6.12.2	Vendor Attributes .....	6-68
6.12.3	Examples .....	6-68
6.13	Model Textures .....	6-69
6.13.1	Handling of Multi-textures .....	6-69
6.13.1.1	Base Texture Layer .....	6-69
6.13.1.2	Subordinate Texture Layer .....	6-70
6.13.1.3	Texture Mapping Conventions .....	6-71
6.13.2	Default Gamma Corrections .....	6-72
6.13.3	Texture Dimension .....	6-72
6.13.3.1	Texture Mipmap .....	6-72
6.13.3.2	Texture Size .....	6-73
6.13.3.3	Texel Size .....	6-73
6.13.4	Texture Palette .....	6-73
6.13.4.1	MModel Example .....	6-73
6.13.4.2	GTModel Example .....	6-73
6.13.4.3	GModel Example .....	6-74
6.13.4.4	T2DModel Example .....	6-74
6.13.5	Usages .....	6-75
6.13.5.1	Model Shadow Textures .....	6-75
6.13.5.2	Model Skin Textures .....	6-77
6.13.5.3	Model Night Maps .....	6-80
6.13.5.4	Model Light Maps .....	6-82
6.13.5.5	Model Tangent-space Normal Maps .....	6-85
6.13.5.6	Model Detail Texture Maps .....	6-86
6.13.5.7	Model Contaminant and Skid Mark Textures .....	6-87
6.13.5.8	Model Cubic Reflection Maps .....	6-87
6.13.5.9	Model Gloss Maps .....	6-89
6.13.5.10	Model Material Textures .....	6-89
6.14	Model Descriptor (Metadata) Datasets .....	6-90
6.14.1	Model Name .....	6-90
6.14.2	Model Identification .....	6-91

6.14.2.1	Moving Model Identification .....	6-91
6.14.2.2	Cultural Feature Identification .....	6-91
6.14.3	Model Mass .....	6-91
6.14.4	Model Parts .....	6-92
6.14.5	Model Textures .....	6-92
6.14.5.1	Texture Metadata .....	6-93
6.14.5.2	Texture Switch .....	6-94
6.14.6	Model Configurations .....	6-95
6.14.6.1	Defining Stations in a Configuration .....	6-95
6.14.6.2	Defining Equipment in a Station .....	6-95
6.14.6.3	Defining Equipment Names .....	6-96
6.14.7	Model Composite Materials .....	6-97
<b>7</b>	<b>CDB Radar Cross Section (RCS) Models .....</b>	<b>7-1</b>
7.1	Introduction .....	7-1
7.2	RCS Data Model .....	7-1
7.2.1	RCS Model Structure .....	7-1
7.3	RCS Polar Diagram Data Representation using Shapefile .....	7-2
7.3.1	Shapefile Internal Data Structure .....	7-2
7.3.1.1	RCS Model Class-Level Attributes: .....	7-6
7.3.1.2	RCS Instance-Level Attribute Data .....	7-11
7.3.2	Multi-Variant RCS Model Applicability .....	7-13
7.3.3	Model's Articulations Effect on RCS Data .....	7-34
<b>8</b>	<b>Glossary .....</b>	<b>8-1</b>
<b>9</b>	<b>Acronyms and Abbreviations .....</b>	<b>9-1</b>
<b>10</b>	<b>Reference Documents .....</b>	<b>10-1</b>
<b>11</b>	<b>List of Contributors .....</b>	<b>11-1</b>



## List of Figures

Figure P-1: Current Approach to Synthetic Environment Generation.....	4
Figure P-2: CDB Approach to Synthetic Environment Generation.....	6
Figure 1-1: Use of CDB as an Off-line Database Repository.....	1-6
Figure 1-2: SE Workflow with CDB as an Off-line Database Repository .....	1-7
Figure 1-3: Use of CDB as an Off-line and On-line Database Repository.....	1-10
Figure 1-4: SE Workflow with CDB as Combined Off-line/Runtime Database Repository ....	1-11
Figure 1-5: Variable Allocation of LOD .....	1-18
Figure 1-6: Typical Evolution of a Database.....	1-23
Figure 1-7: Typical Implementation of CDB Specification on High-end Simulator.....	1-25
Figure 1-8: Typical Implementation of CDB Specification on Desktop Computer .....	1-26
Figure 1-9: Pipelined DB Update Process .....	1-29
Figure 1-10: CDB Specification Tile/Layer Structure.....	1-35
Figure 1-11: Typical CDB Implementation on a Suite of Simulators .....	1-37
Figure 1-12: Typical DB Generation - CDB Used as DB Repository .....	1-38
Figure 1-13: Typical DB Generation Flow - CDB Used as DB & Sim Repository .....	1-40
Figure 1-14: Versioning Paradigm Applied to Dynamic SE .....	1-48
Figure 1-15: Sources of Synthetic environment Database Correlation Errors .....	1-53
Figure 1-16: Overload Limit of Least Capable Client-Device .....	1-55
Figure 1-17: Overload Limit of Each Client-Device .....	1-56
Figure 1-18: Operating Limit of Least Capable Client-Device .....	1-56
Figure 1-19: Individually Adjusted to the Operating Limit of Each Client-Device .....	1-57
Figure 1-20: Adjusted by the Simulator Operator at Scenario Startup .....	1-57
Figure 2-1: CDB Earth Slice Representation.....	2-3
Figure 2-2: Variation of Longitudinal Dimensions versus Latitude .....	2-4
Figure 2-3: Tile-LOD Hierarchy.....	2-8
Figure 2-4: Earth Slice Example (Five Levels-of-Detail).....	2-10
Figure 2-5: Extract from Light Naming Hierarchy .....	2-15
Figure 2-6: Seabed Composite Material .....	2-20
Figure 2-7: Oil Tank Composite Materials.....	2-20
Figure 2-8: Thermal Simulation of Oil Tank Composite Materials .....	2-21
Figure 2-9: Flow of Material Attribution Data .....	2-24
Figure 2-10: Linking CDB Base Materials to SEM Base Materials.....	2-25
Figure 2-11: SEM Base Material Properties Table.....	2-25
Figure 3-1: UML Diagram of CDB Version Concept .....	3-5
Figure 3-2: A Valid Chain of CDB Versions.....	3-5
Figure 3-3: UML Diagram of CDB Extension Concept.....	3-6
Figure 3-4: Adding content to the CDB.....	3-7
Figure 3-5: Modifying Content of the CDB .....	3-8
Figure 3-6: Deleting Content from the CDB .....	3-8
Figure 3-7: UML Diagram of CDB Configuration Concept .....	3-9
Figure 3-8: Allocation of CDB Geocells with Increasing Latitude .....	3-50
Figure 5-1: Center Grid Data Elements .....	5-31
Figure 5-2: Corner Grid Data Elements.....	5-31

Figure 5-3: Center Data Elements as a Function of LODs .....	5-32
Figure 5-4: Corner Data Elements as a Function of LODs.....	5-32
Figure 5-5: Example of Digital Elevation Model (DEM).....	5-33
Figure 5-6: DEM Depicted as a Grid of Elevations at Regular Sample Points .....	5-34
Figure 5-7: CDB Mesh Types.....	5-35
Figure 5-8: Primary Terrain Elevation Component.....	5-37
Figure 5-9: Modeling of Wells, Overhanging Cliffs and Tunnels.....	5-38
Figure 5-10: Encoding of Lat/Long Offsets.....	5-40
Figure 5-11: Grid Element Coverage within a CDB Tile .....	5-40
Figure 5-12: Storage Tank Point-Feature .....	5-44
Figure 5-13: Road Lineal Feature .....	5-45
Figure 5-14: LOD Structure of Raster Datasets.....	5-46
Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D) .....	5-48
Figure 5-16: Generation of LODs for the MinMaxElevation Dataset (2D) .....	5-49
Figure 5-17: Generation of LODs for the MinMaxElevation Dataset (1D) – Special Case.....	5-50
Figure 5-18: Availability of LODs for Elevation and MinMaxElevation Datasets.....	5-51
Figure 5-19: Missing MinMaxElevation Datasets.....	5-54
Figure 5-20: Primary Terrain Elevation and Subordinate Bathymetry Components .....	5-57
Figure 5-21: Derived Earth Elevation Values.....	5-58
Figure 5-22: Example of Primary Terrain Elevation and Bathymetry Components .....	5-58
Figure 5-23: Terrain Elevation, Bathymetry and Tide Components .....	5-62
Figure 5-24: Derived Earth Elevation, Water Elevation and Surface Elevation Values .....	5-62
Figure 5-25: Projection of Terrain Imagery Dataset onto Terrain Elevation Dataset.....	5-73
Figure 5-26: Image Classification Example .....	5-77
Figure 5-27: Example of a Raster Material Dataset.....	5-78
Figure 5-28: Instance-level Attribution Schema.....	5-90
Figure 5-29: Class-level Attribution Schema .....	5-91
Figure 5-30: Relation between Shapes and Attributes.....	5-94
Figure 5-31: RTAI Typical Usage Histogram .....	5-126
Figure 5-32: Example of a Topological Network .....	5-139
Figure 5-33: Example of International Boundaries .....	5-152
Figure 5-34: Example of City Locations .....	5-153
Figure 5-35: Example of State Capital Locations and Time Zone Boundaries .....	5-153
Figure 6-1: General OpenFlight Tree Structure.....	6-2
Figure 6-2: Internal Structure of CDB Models.....	6-3
Figure 6-3: Internal Structure of T2DModels .....	6-4
Figure 6-4: Typical Structure of a Model Master File.....	6-7
Figure 6-5: Model Coordinate System.....	6-9
Figure 6-6: Coordinate System – Aircraft .....	6-10
Figure 6-7: Coordinate System – Helicopter .....	6-10
Figure 6-8: Coordinate System – Ship.....	6-11
Figure 6-9: Coordinate System – Ground-based Model .....	6-11
Figure 6-10: Coordinate System – Lifeform.....	6-12
Figure 6-11: Coordinate System – Cultural Feature .....	6-12
Figure 6-12: Coordinate System – Power Pylon.....	6-13

Figure 6-13: Model Global Zone .....	6-18
Figure 6-14: Simple Zone .....	6-20
Figure 6-15: Articulated Zone .....	6-20
Figure 6-16: Zone Hierachy .....	6-21
Figure 6-17: Simple Zone Graphical Representation .....	6-21
Figure 6-18: Additive LOD to Control the Graphical Representation .....	6-22
Figure 6-19: Exchange LODs to Select the Graphical Representation.....	6-22
Figure 6-20: Switch Node to Select the Graphical Representation .....	6-23
Figure 6-21: Footprint Zone Structure.....	6-26
Figure 6-22: Cutout Zone Structure.....	6-27
Figure 6-23: Model Shell Structure .....	6-28
Figure 6-24: Model Interior Structure.....	6-29
Figure 6-25: Interior Zone Structure.....	6-30
Figure 6-26: Floor Zone Structure .....	6-33
Figure 6-27: Room Zone Structure .....	6-34
Figure 6-28: Fixture Zone Structure .....	6-35
Figure 6-29: Fixture Zone Structure .....	6-36
Figure 6-30: Partition Zone Structure.....	6-37
Figure 6-31: Aperture Structure.....	6-38
Figure 6-32: Surface Zone Structure.....	6-38
Figure 6-33: Orientation of the Chinook Helicopter.....	6-40
Figure 6-34: The Body of the Chinook Helicopter.....	6-41
Figure 6-35: The DIS Origin of the Chinook Helicopter.....	6-42
Figure 6-36: Conforming Vertices to Terrain.....	6-44
Figure 6-37: Origin Conformal Mode.....	6-46
Figure 6-38: Vertex Conformal Mode Example .....	6-47
Figure 6-39: Line Conformal Mode.....	6-48
Figure 6-40: Plane Conformal Mode .....	6-49
Figure 6-41: Application of Line and Plane Conformal Modes on 3D Roads .....	6-50
Figure 6-42: Surface Confomal Mode .....	6-51
Figure 6-43: Exchange and Additive LOD Nodes.....	6-53
Figure 6-44: Exchange LOD Nodes .....	6-54
Figure 6-45: General Damage State Tree Structure.....	6-60
Figure 6-46: Damage States Ordering .....	6-61
Figure 6-47: Example of a Texture Representing a Rotor .....	6-62
Figure 6-48: Multiple Versions of Rotating Parts.....	6-63
Figure 6-49: Using Shadow Polygons .....	6-75
Figure 6-50: Example of a Shadow Map in the XY Plane .....	6-76
Figure 6-51: The M1A2 Abrams with a Desert Camouflage .....	6-77
Figure 6-52: The M1A2 Abrams with a Forest Camouflage.....	6-78
Figure 6-53: M1A2 Desert Skin Mosaic.....	6-79
Figure 6-54: Base Texture .....	6-81
Figure 6-55: Night Map .....	6-81
Figure 6-56: Light Map.....	6-83
Figure 6-57: Combined Effect of Base Textures and Light Maps.....	6-84



Figure 6-58: Combined Effect of Night and Light Maps .....	6-84
Figure 6-59: Normal Map Sample .....	6-85
Figure 6-60: Detail Texture Map Sample .....	6-86
Figure 6-61: Environment Used to Produce Reflection Map .....	6-87
Figure 6-62: Resulting Reflection Map .....	6-88
Figure 6-63: Rendered Reflection Map onto Reflecting Cube .....	6-88
Figure 7-1: Graphical Representation of the 3D Model RCS Shape Data .....	7-3
Figure 7-2: Polar Diagram of RCS Data in Planar Representation .....	7-4
Figure 7-3: Polar Diagram of RCS Data in Spherical Representation.....	7-5
Figure 7-4: UML Representation of the 3D Model RCS Shapefile Structure .....	7-12
Figure 7-5: Example of RCS Shapefile .....	7-13



## List of Tables

Table 1-1: Summary of Synthetic Environment Database Correlation Errors .....	1-51
Table 2-1: Intervals for DTED Level 2.....	2-2
Table 2-2: Size of CDB Geocell per Zone.....	2-2
Table 2-3: CDB Geocell Unit Size in Arc Seconds.....	2-5
Table 2-4: CDB LOD vs Tile and Grid Size.....	2-6
Table 2-5: Character Set Used for CDB Files and Folders.....	2-12
Table 2-6: Components of a Base Material .....	2-18
Table 3-1: CDB LOD vs Model Resolution .....	3-16
Table 3-2: GTModelGeometry Directory Structure .....	3-21
Table 3-3: GTModelGeometry Entry File Naming Convention.....	3-22
Table 3-4: GTModelGeometry Level of Detail Naming Convention .....	3-22
Table 3-5: GTModelDescriptor Naming Convention.....	3-23
Table 3-6: GTModelTexture Directory Structure.....	3-24
Table 3-7: GTModelTexture Naming Convention .....	3-25
Table 3-8: GTModelMaterial Naming Convention .....	3-25
Table 3-9: GTModelMaterial Naming Convention .....	3-26
Table 3-10: GTModelInteriorGeometry Directory Structure .....	3-28
Table 3-11: GTModelInteriorGeometry Naming Convention.....	3-29
Table 3-12: GTModelInteriorDescriptor Naming Convention.....	3-29
Table 3-13: GTModelInteriorTexture Directory Structure.....	3-31
Table 3-14: GTModelInteriorTexture Naming Convention .....	3-31
Table 3-15: GTModelInteriorMaterial Naming Convention .....	3-32
Table 3-16: GTModelSignature Directory Structure.....	3-33
Table 3-17: GTModelSignature Naming Convention .....	3-35
Table 3-18: MModelGeometry Directory Structure .....	3-37
Table 3-19: MModelGeometry Naming Convention .....	3-37
Table 3-20: MModelDescriptor Naming Convention.....	3-38
Table 3-21: MModelTexture Directory Structure.....	3-39
Table 3-22: MModelTexture Naming Convention .....	3-39
Table 3-23: MModelMaterial Naming Convention.....	3-40
Table 3-24: MModelMaterial Naming Convention.....	3-40
Table 3-25: MModelSignature Directory Structure.....	3-41
Table 3-26: MModelSignature Naming Convention .....	3-42
Table 3-27: CDB LOD versus Feature Density .....	3-44
Table 3-28: Tiled Dataset Directory Structure.....	3-47
Table 3-29: NbSliceIDIndex for every CDB Zones .....	3-50
Table 3-30: Tiled Dataset File Naming Convention 1 .....	3-54
Table 3-31: GTModelGeometry Entry File Directory Structure .....	3-58
Table 3-32: NavData Naming Convention .....	3-58
Table 4-1: CDB File Format Compatibility.....	4-3
Table 5-1: Component Selectors for Metadata Datasets.....	5-1
Table 5-2: Component Selectors for Navigation Dataset .....	5-15
Table 5-3: List of Navigation Components .....	5-16

Table 5-4: List of Navigation Schema Attributes .....	5-19
Table 5-5: Example of a Navigation Schema .....	5-19
Table 5-6: List of Navigation Key Attributes .....	5-21
Table 5-7: Example of Navigation Keys .....	5-21
Table 5-8: Component Selectors for CDB Model Textures .....	5-23
Table 5-9: Component Selectors for GTModel Datasets.....	5-24
Table 5-10: Component Selectors for Mmodel Datasets .....	5-26
Table 5-11: Elevation Dataset Components .....	5-36
Table 5-12: Partial List of Hypsography FACCs (for Offline Terrain Constraining) .....	5-42
Table 5-13: List of Hypsography FACCs (for Online Terrain Constraining) .....	5-43
Table 5-14: XML Tags for the JPEG 2000 Metadata.....	5-71
Table 5-15: Imagery Dataset Components .....	5-72
Table 5-16: VSTI Default Read Values.....	5-74
Table 5-17: Raster Material Dataset Components .....	5-79
Table 5-18: Component Selector 2 for Vector Datasets .....	5-84
Table 5-19: Boundary Type Enumeration Values .....	5-99
Table 5-20: Location Accuracy Enumeration Values.....	5-113
Table 5-21: Location Type Enumeration Values.....	5-115
Table 5-22: Populated Place Type Enumeration Values .....	5-125
Table 5-23: Surface Roughness Enumeration Values .....	5-129
Table 5-24: Structure Shape Category Enumeration Values .....	5-131
Table 5-25: Structure Shape of Roof Enumeration Values .....	5-134
Table 5-26: Urban Street Pattern Enumeration Values.....	5-136
Table 5-27: Allocation of CDB Attributes to Vector Datasets .....	5-144
Table 5-28: Tiled Navigation Dataset.....	5-148
Table 5-29: Component Selectors for GSFeature Dataset.....	5-149
Table 5-30: Component Selectors for GTFeature Dataset.....	5-150
Table 5-31: Component Selectors for GeoPolitical Feature Dataset .....	5-151
Table 5-32: Component Selectors for RoadNetwork Dataset.....	5-155
Table 5-33: Component Selectors for RailRoadNetwork Dataset.....	5-157
Table 5-34: Component Selectors for PowerLineNetwork Dataset .....	5-158
Table 5-35: Component Selectors for HydrographyNetwork Dataset.....	5-159
Table 5-36: Vector Composite Material Table Component .....	5-160
Table 5-37: Component Selectors for GSModel Datasets .....	5-161
Table 5-38: Component Selectors for T2DModel Datasets.....	5-162
Table 6-1: Sample XML Tag Used in a Comment Record .....	6-6
Table 6-2: XML Tags for Zones .....	6-18
Table 6-3: OpenFlight Records for a Zone .....	6-18
Table 6-4: XML Tags for Hot Spots.....	6-20
Table 6-5: Footprint Zone XML Tags .....	6-26
Table 6-6: XML Tags for Landing Zones.....	6-27
Table 6-7: Shell Zones XML Tags .....	6-29
Table 6-8: Interior Zone XML Tags .....	6-29
Table 6-9: UHRB Class Names and corresponding CDB Zone Names .....	6-31
Table 6-10: XML Tags for Zone Connections .....	6-31

Table 6-11: Examples of Absolute and Relative Paths.....	6-32
Table 6-12: Floor Zone XML Tags .....	6-33
Table 6-13: Room Zone XML Tags .....	6-34
Table 6-14: Fixture Zone XML Tags.....	6-35
Table 6-15: Partition Zone XML Tags .....	6-36
Table 6-16: Aperture Zone XML Tags .....	6-37
Table 6-17: XML Tags for Points.....	6-39
Table 6-18: OpenFlight Records for a Point.....	6-39
Table 6-19: XML Tags for the DIS Origin.....	6-40
Table 6-20: XML Tags for a Viewpoint .....	6-43
Table 6-21: XML Tags for Attach Point.....	6-43
Table 6-22: XML Tags for Anchor Point .....	6-43
Table 6-23: XML Tags for Center of Mass .....	6-44
Table 6-24: Conformal Modes.....	6-45
Table 6-25: Maximum Number of Vertices per Model-LOD .....	6-57
Table 6-26: XML Tags to Create a CDB Switch.....	6-59
Table 6-27: OpenFlight Records to Create a CDB Switch .....	6-59
Table 6-28: XML Tags for Damage State Switch .....	6-60
Table 6-29: Example of a Damage State Switch with Two Transitions.....	6-60
Table 6-30: XML Tags for Motion Blur Switch.....	6-63
Table 6-31: Example of a Motion Blur Switch with One Transition .....	6-64
Table 6-32: XML Tags for DOF.....	6-64
Table 6-33: OpenFlight Records for a Light Point .....	6-67
Table 7-1: ModelSignature Significant Angle per LOD.....	7-4
Table 7-2: XML Tags for Hot Spots.....	7-6
Table 7-3: RCS Instance Attribute Fields .....	7-12
Table 7-4: Radar Model Numbers .....	7-13



## Preface

### Intended Audience

The primary audience for this document includes distributed simulation system developers and synthetic environment database tool developers whose applications are intended to read and/or write synthetic environment database files. To this end, this document discusses concepts incorporated in the Specification and contains a detailed description of the physical layout of the files as represented on disk.

This document assumes the reader is familiar with:

- (1) Synthetic environment creation and generation process.
- (2) Existing database interchange and database visualization standards (such as SEDRIS, SIF, OpenFlight, DIS, DIGEST, Shapefile, TIFF, JPEG, etc.).
- (3) Image Generation and radar simulation principles.
- (4) Sensor subsystems used in aircraft and other vehicles (Radar, Night Vision Image Intensifiers, Infrared (IR) sensors, Laser Range Finders, etc.) requiring the use of synthetic environment databases.
- (5) Simulator client-devices (such as Computer Generated Forces, Air/Ground Traffic Control, Weather, etc.) requiring the use of synthetic environment databases.
- (6) Principles of Object-Oriented (OO) programming.

### Problem Definition

Complex mission simulators include a wide range of subsystems designed to simulate on-board equipment and to provide a rich gaming environment complete with weather, computer generated forces, ordnance, air traffic, networked players, etc. Each of these subsystems typically utilizes a proprietary runtime database (and format) for its synthetic representation of the gaming area. Traditionally, these application-specific formats have been generated off-line at a database generation facility using a variety of tools and processes. This approach has several inherent disadvantages, including length of time needed for synthetic environment production for multiple simulation applications, loss of correlation due to compilation differences, complexity in configuration management, and an inefficient update process. The abundance of distinct database formats creates several challenges for configuration management, resulting in mismatched correlation of the various cues, and in the increased timeline needed to generate these databases. The CDB Specification is a new approach that establishes a set of industry-standard formats and conventions for all simulator client subsystems (aka simulator client-devices).

Digital computer based flight simulation dates back to the mid-60s. Many legacy processes and assumptions have predicated the creation, maintenance, and modification of traditional military tactical training flight simulator synthetic environment databases (DBs). Early approaches were usually constrained by severe

hardware, software and data source limitations. This in turn would force simulation engineers to make important compromises between a subsystem's targeted fidelity and its level of generality, scalability, abstraction, and correlation with other simulator client-devices. Industry wide standardization could not be readily achieved because technologically viable options only offered partial solutions to these needs. Digital technologies have made tremendous strides in the past 10 years and are narrowing the “gap” between what is required for training and what the technology can now deliver. As this trend continues, simulation engineers can re-examine all of these earlier trade-offs, and redress past compromises.

The approach to visual system synthetic environments is rife with such compromises. For instance, most DB formats in use today still require a full off-line re-compilation of the DB into a (usually proprietary) runtime format, even for a small-area update. As a result, the creation and update of such databases is still a recurring labor-intensive exercise. Separate, non-harmonized, Image Generator (IG) manufacturer proprietary tools are required to generate and modify the DB, resulting in sometimes incongruous, incomplete, and closed DB formats. The typical evolution of the DB format poses important configuration control challenges due to the required reconciliation of incompatible revisions of these DB formats. The entire process is further aggravated by the time-consuming off-line compilation of visual, sensor, threat, and air traffic DBs.

The confluence of digital multi-spectral high-resolution satellite imagery and highly capable visual systems has created dramatic new Mission Planning and Mission Rehearsal capabilities. As a result, recent environment databases built to take advantage of these new capabilities require orders of magnitude more storage than equivalent databases just a few years ago.

It is clear that, if left unchecked, these factors will become important cost drivers and currently impact simulators (see Figure P-1: Current Approach to Synthetic Environment Generation) in the following areas:

(1) Size of the Synthetic Environment Storage:

In order to satisfy the runtime device loadable representation of each of the simulator client-devices, the synthetic environment is replicated several times. Most runtime device-loadable formats do not take advantage of modern compression schemes. Furthermore, each of runtime device-loadable formats imposes its own distinct database structure; in many cases, the organization of the runtime database is not sufficiently flexible to permit small-area changes without a recompilation of the database. Each recompilation generates a complete copy of the database that must then be deployed and put under configuration control.

(2) Longevity of the Synthetic Environment Database:

The runtime loadable representation of each of the simulator client-devices is closely associated with the fidelity, the features, and the format restrictions imposed by the simulator client-devices. Increasingly, the design of simulator

client-devices is being influenced by the rapidly evolving Commercial-Off-The-Shelf (COTS) technology. However, few of the benefits of the new simulator technology can be realized unless the runtime DB is rebuilt to the new simulator capabilities.

(3) Scalability of the Synthetic Environment Database:

The runtime loadable representation typically provided by current vendors for each of the simulator client-devices is closely aligned to the total content and density limits imposed by each simulator client-devices. Once “frozen” into the runtime loadable DB, it is difficult to fully take advantage of emerging simulator technologies capable of handling greater DB content/density.

(4) Environment Database Correlation:

Correlation between simulator client-devices is aggravated due to the alternate data representations demanded by each of the (often proprietary) simulator client-devices. Data representations can vary in resolution, precision, and in fidelity. A solution to the correlation issue is costly because each of the runtime DBs must be re-compiled to the capabilities of the most limited simulator client-device.

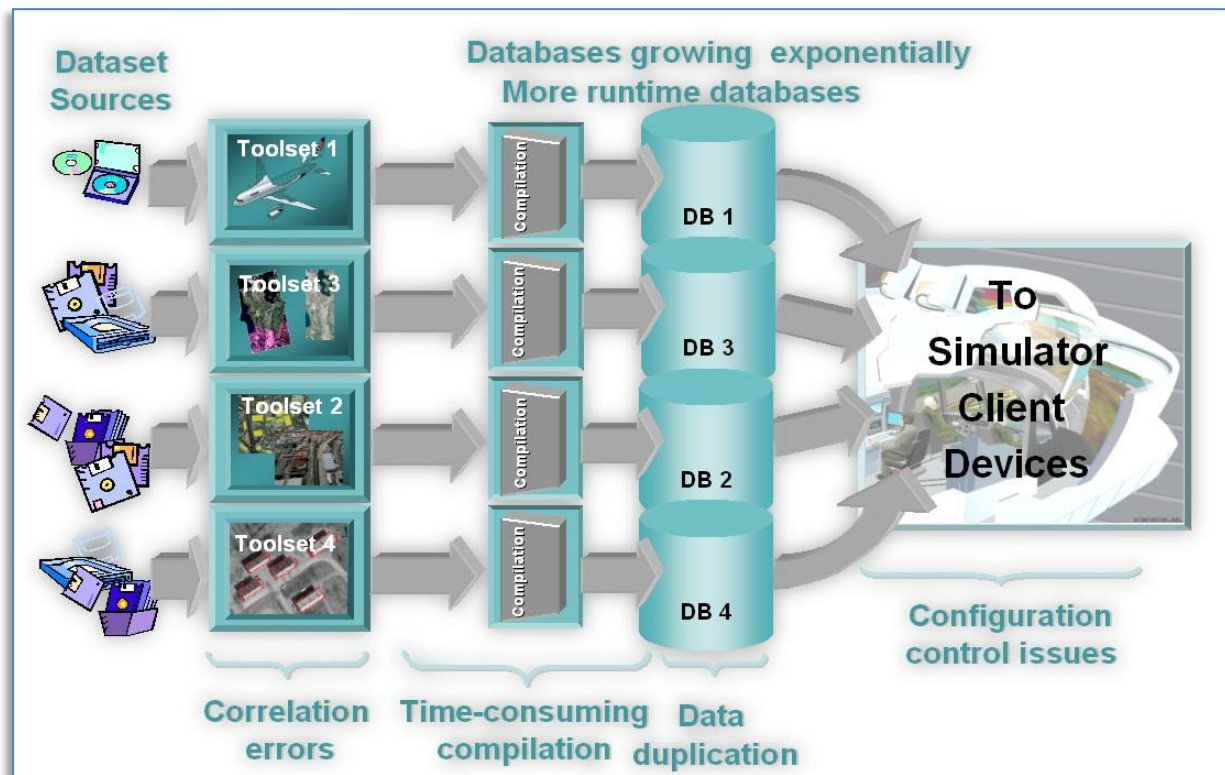
(5) Database Availability Timeline:

The extensive off-line compilation process that produces the runtime databases is time-consuming; furthermore, this is aggravated because important amounts of data are replicated in each of the runtime device-specific databases. While a parallel processing approach could alleviate this, many of current database compilation tools are not capable of supporting this. The transfer of the replicated databases from the DB generation facility to the simulator(s) wastes additional time. Finally, small-area updates are time-consuming because of the monolithic structure of the client-device runtime DBs.

(6) Configuration management:

The configuration management of distinct simulator client-devices requires additional effort because the client-specific runtime DBs must each be re-derived and re-compiled from the raw source data. The evolution of the (often proprietary) runtime DB formats poses additional configuration control challenges because prior versions of the format cannot be fully reconciled with new ones. Hence, backward compatibility cannot always be assured. Finally, the runtime DBs are often massive entities; small area-updates cannot be undertaken without re-compiling the entire DB.





**Figure P-1: Current Approach to Synthetic Environment Generation**

The CDB Specification addresses these and other shortcomings through a common database Specification. It is intended as a simulation Specification for use in producing a unified synthetic representation of the world. A database built to the CDB Specification is referred to as a Common Database (CDB). A CDB is a single-copy data repository from which various simulator client-devices are able to simultaneously retrieve, in real-time, relevant information to perform their respective runtime simulation tasks.

The CDB Specification enhances unity and correlation between various simulator level client-devices (e.g., Visual, Radar), while improving database maintainability. As a result, one of the main benefits of the CDB Specification is the elimination of several types of correlation errors attributable to alternate, sometimes conflicting data representations required by each the simulator client-devices. The Specification achieves this by allowing all simulator clients-devices to share, in runtime, a single repository of the synthetic environment information. In addition, a CDB can also be used as a master repository for the authoring tools; as a result CDB content can be interchanged between DB workstations. Finally, in the case where one or more of the client-devices are not compliant to this CDB Specification, it is possible to revert to the conventional compilation paradigm, (i.e., compile the CDB into one or more client-device loadable (usually proprietary) representations).

The CDB Specification internal data representation model is based on the native formats used by industry-standard toolsets. As a result, it eliminates the time-



consuming off-line database compilation process for each of the clients. The CDB Specification redefines a new balance between off-line and on-line compilation processes because modern computer platforms can now accomplish most of the compilation process in real-time<sup>1</sup>.

The CDB Specification addresses the issues that have characterized the simulation industry for past decades (see Figure P-2: CDB Approach to Synthetic Environment Generation), as follows:

(1) Size of Synthetic Environment Storage:

The CDB consolidates the synthetic environment into a single data repository that provides a static representation of the earth. It includes all the relevant information so that all the simulator client-devices can perform their respective simulation tasks in order to meet the training and mission rehearsal requirements. It avoids any data content duplication. Storage intensive datasets can be optionally compressed using modern third-party algorithms. The CDB Specification provides a fine-grain versioning scheme that avoids the replication of the entire DB when effecting small-area updates.

(2) Scalability of Synthetic Environment Database:

A CDB can be built to a size or a density that far exceeds the capabilities of some or all of its client-devices. The data structure of the CDB Specification makes it possible to implement runtime filtering schemes to adjust the loaded CDB content density to the specific capabilities of the client-devices. As a result, a CDB can be scaled to take advantage of future simulator technological improvements.

(3) Environment DB Correlation:

Since CDB content is unique (without data duplication), runtime source level correlation errors among clients are eliminated, thereby ensuring inter-subsystem coherence and simulator interoperability. In addition, it is possible to improve correlation by adjusting the runtime publishing process associated with each client-device.

(4) Database Availability Timeline:

The CDB generation process allows for small database incremental updates, thereby shortening generation and build process times. Furthermore, the translation step into CDB format is rather straightforward since the CDB Specification is based on industry-standard native tool formats.

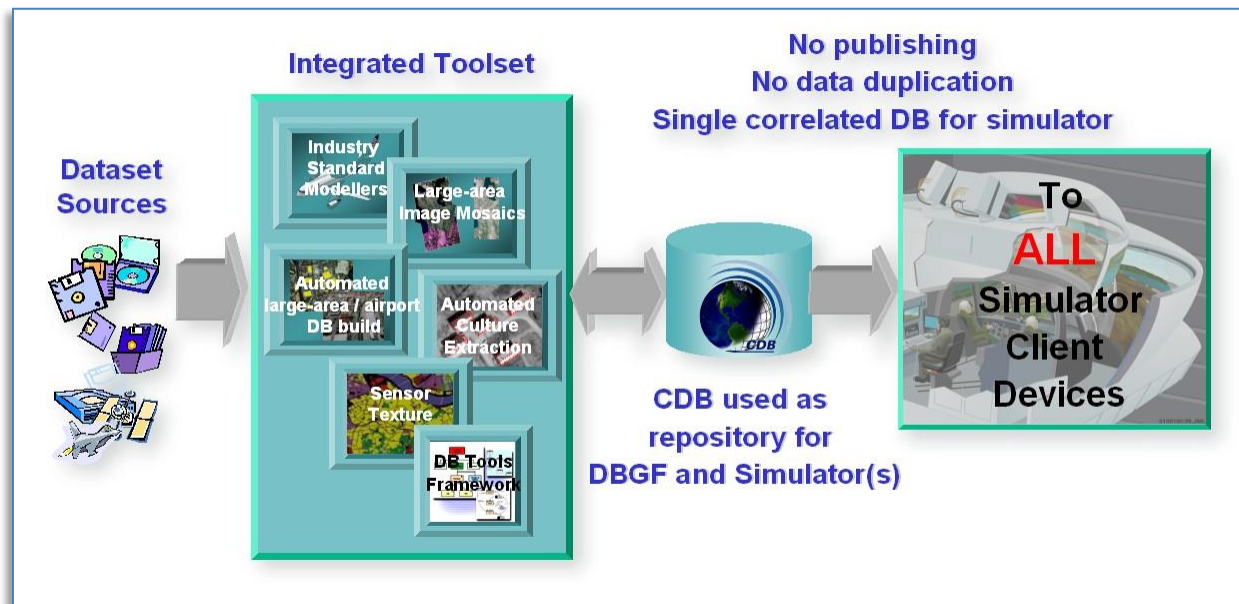
(5) Configuration Management:

Configuration management effort is reduced, because a single CDB corresponds to the synthetic environment of all the client-devices in the simulator. Furthermore, while a CDB is conceptually a single, yet layered

---

<sup>1</sup> The CDB Specification does require however, that most of its dataset be generated in a level-of-detail hierarchy.

entity, the Specification internally supports incremental updates resulting in efficient storage and handling of CDB versions.



**Figure P-2: CDB Approach to Synthetic Environment Generation**

## Chapter 1

### 1 Introduction

#### 1.1 Purpose

This Specification provides a full description of a data model (aka schema) for the synthetic representation of the world. The representation of the synthetic environment in CDB format 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 Specification provides a Common Environment to which inter-connected simulators share a common view of the simulated environment.

#### 1.2 Document Structure

A significant portion of the CDB Specification concerns itself with aspects of the data model that relate to the structure of the database 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. All aspects of the CDB structure are covered in Chapter 3.

A second important aspect of the CDB Specification deals with all of the naming conventions that are internal to the CDB. Names provide the simulator client-devices the necessary means to understand the meaning of and to control the various elements modeled within the synthetic environment. For example, the CDB naming conventions provide all the simulator client-devices the means to control cultural lighting, to articulate a landing gear on an aircraft, to animate a trailing wake on a ship, to control the heat emitted by a tank engine, to position a car on the ground, etc. Names also provide the means to attribute lights and to attribute materials. All of the CDB Concepts are stated in Chapter 2 that also deals with the naming and handling of materials that make up the synthetic environment.

Chapter 4, CDB File Formats provides a description of all the formats prescribed by the CDB Specification.

Chapter 5, CDB Datasets provides a detailed description of all CDB datasets.

OpenFlight plays a significant role within the CDB Specification since all of the statically positioned cultural features and the moving models are represented in this format. However, for it to be truly useful in the context of simulation, the OpenFlight format must be supplemented with a set of agreed-upon conventions that can be used by all client-devices and the simulation software. Chapter 6, CDB OpenFlight Models establishes all such OpenFlight conventions for the CDB Specification.

For devices such as Radars, a geometric representation of a model may often provide a level of fidelity which is insufficient or inappropriate for use in simulation or alternately, it may not be feasible to compute a radar cross-section (RCS) of the model in real-time. Alternately, a user may wish to incorporate real-world RCS data into the simulator client-devices in order to further improve simulation fidelity. To this end, the CDB Specification defines a RCS (Radar Cross-Section) model representation for use by Sensor Simulation client-devices such as Radar and/or Sonar. Chapter 7, CDB Radar Cross Section (RCS) Models establishes a set of conventions that permit RCS representations using the Shapefile format.

The CDB Specification relies heavily on five established industry formats, namely the TIFF format (Appendix B), the OpenFlight format (Appendix C), the RGB format (Appendix P), the Shapefile format (Appendix D) and the JPEG 2000 file format (Appendix T). These Specifications have been included as appendices to this Specification. Each of these documents has been annotated to reflect the conventions established by the CDB Specification. The conventions define how TIFF, OpenFlight, RGB, Shapefile and JPEG 2000 formatted files are to be interpreted by CDB-compliant simulator readers.

Appendices E and F provide the CDB light type naming hierarchy and the CDB model component hierarchies respectively while Appendix L provides the material list for the CDB Specification.

Other Appendices further describes other aspects of the CDB Specifications like providing the CDB Directory Naming and Structure (Appendix M), the mapping of FACC Codes (Appendix N), the List of Texture Component Selectors (Appendix O), the SGI Image File Format (Appendix P), the Table of Dataset Codes (Appendix Q) or how some datasets are derived from others (Appendix R).

## **1.3 Scope**

The Specification defines an earth synthetic environment data model and the representation, organization, storage structure and conventions necessary to support all of the subsystems of a full-mission simulator. The Specification makes use of several commercial and simulation data formats endorsed by leaders of the database tools industry.

The CDB synthetic environment is a representation of the natural environment including external features such as man-made structures and systems. It encompasses the 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 synthetic environment includes the specific attributes of the synthetic environment data as well as their relationships.

A database that conforms to the CDB Specification (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 synthetic environmental data in a CDB is specifically tailored for real-time applications.

### 1.3.1 What is the CDB Specification

The CDB Specification is an open synthetic environment database Specification to which the U.S. Government has unrestricted rights. The CDB Specification is rooted in a group of formats well established within the simulation industry. To each of these formats, the CDB Specification provides a comprehensive set of conventions appropriate to the field of simulation. The Specification defines all aspects of data representation and organization, storage structure to support full-mission simulation. A database that conforms to the CDB Specification (i.e., a CDB) contains datasets organized in layers and tiles that represent the features of a synthetic environment for the purposes of distributed simulation applications. A CDB can be readily used by existing simulation client-devices (legacy IGs, Radars, CGF, etc.) through a publishing process performed in real-time. The data structures used in CDB Specification synthetic environment databases are different than those used in relational databases mostly because the CDB has chosen to standardize on formats adopted by the simulation community. This facilitates the work required to adapt existing authoring tools to read/write/modify the CDB and the task to develop runtime publishers (RTP) designed to operate on these data structures.

The CDB Specification is fundamentally about:

1. A representation of the natural earth and man-made synthetic environment for the field of simulation.
2. A turnkey, as-is representation of the Synthetic Environment (SE) for use in real-time distributed simulation.

The synthetic environment is a representation of the natural environment at a specific geographical location including the external features of the man-made structures and systems. Therefore, the synthetic environment includes the terrain, the terrain features (both natural and man-made), three-dimensional (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 synthetic environment includes the specific attributes of the synthetic environment data as well as their relationships. The CDB Specification is more than just a means of creating visual (aka out-the-window) scenery. Unlike other Specifications that only deal with data representational types of polygons, colors, and textures, it deals with all the data representational types needed in high-end virtual and constructive simulation applications.

The bulk of the CDB internal data representation is based on five commercial data formats endorsed by leaders of the simulation database tools industry, namely:

***TIFF/GeoTIFF:*** for the representation of terrain altimetry, terrain surface characteristics relevant to simulation.

***OpenFlight:*** for the representation of 3D culture and moving models.

**RGB:** for the textures associated with 3D culture and moving models.

**Shapefile:** for the instancing and attribution of statically positioned point, lineal and areal 2D/3D culture features.

**JPEG 2000:** for a representation of terrain raster imagery comprising a well defined and accepted compression method that allows both lossy and lossless schemes.

The CDB Specification storage structure allows efficient searching, retrieval and storage of any information contained within the CDB. The storage structure portion of the Specification defines a comprehensive binary file description, (i.e., it specifies the exact format of all files used in the implementation of the Specification). Storage structure aspects include descriptions of each information field used within CDB Specification files, including data types and data type descriptions.

The CDB Specification relies on three important means to organize the environmental data:

1. **Tiles:** The CDB storage structure allows efficient searching<sup>2</sup>, retrieval and storage of any information contained within the CDB. The storage structure portion of the Specification geographically divides the world into geodetic tiles (bound by latitudes and longitudes), each containing a specific set of features (such as terrain altimetry, vectors) and models (such as OpenFlight models, RCS models), which are in turn represented by the datasets (see Figure 1-10: CDB Specification Tile/Layer Structure). The datasets define the basic storage unit used in a CDB. 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
2. **Layers:** The CDB standard data representation 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).
3. **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 ownship. 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 databases having near-infinite content. The CDB standard is structured into an 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.

---

<sup>2</sup> A information is retrieved by means of an implicit index defined by this Specification



The Specification does not define or enforce an operating system or file system; nonetheless, the implementation of a CDB storage sub-system must conform to absolute minimum file system requirements called for by the Specification.

### 1.3.1.1 Use of CDB as an Off-line Database Repository

Figure 1-1: Use of CDB as an Off-line Database Repository, illustrates the deployment process of a CDB when it is used solely as an off-line Master database repository. This approach follows the SE deployment paradigm commonly used today within the simulation community. The use of the CDB as an off-line environmental data repository offers immediate benefits, namely...

- SE Standardization through a public, open, fully-documented database schema that is already supported by several SE authoring tools.
- SE Plug-and-Play Portability and Interoperability across various vendor SE authoring toolsets
- SE Correlation through the elimination of source correlation errors through normalization of all data sets (a single representation for each dataset)
- SE Re-use by eliminating dependencies that are specific to the simulation application, the Database Generation tool suite, the simulation program, the technology
- SE Scalability which results in near-infinite SE addressability, spatial resolution and content density in each of the SE datasets.
- 3D Model Library Management through built-in provisions for the cataloging of models
- SE Versioning Mechanism allowing instant access to prior versions and simplified configuration management
- Cooperative SE Workflow through an internal SE structure which favors team work. The SE workflow can be allocated by specialty (e.g., altimetry, satellite imagery, vector data) or by geographic footprint.
- Straightforward SE Archival and Recovery

Note that the use of the CDB as an off-line repository does not impose any change to the simulation training equipment (i.e., no modifications to client-devices are required<sup>3</sup>). However, the deployment of the synthetic environment is similar to the conventional approaches used in industry requiring the time-consuming, storage-intensive, off-line compilation of proprietary runtime databases to each client-device. Furthermore, the computing demands on the database generation facility are

---

<sup>3</sup> Or alternately, runtime publishers need not be developed for client-devices

significantly greater because the entire database must be published off-line for each client-device before it can be deployed. These costs rapidly escalate with the complexity and size of the synthetic environment, the number of supported client-devices and the number of supported training facilities. For complex databases, these costs can far outweigh the costs of the runtime publishers attached to each simulator client-device.

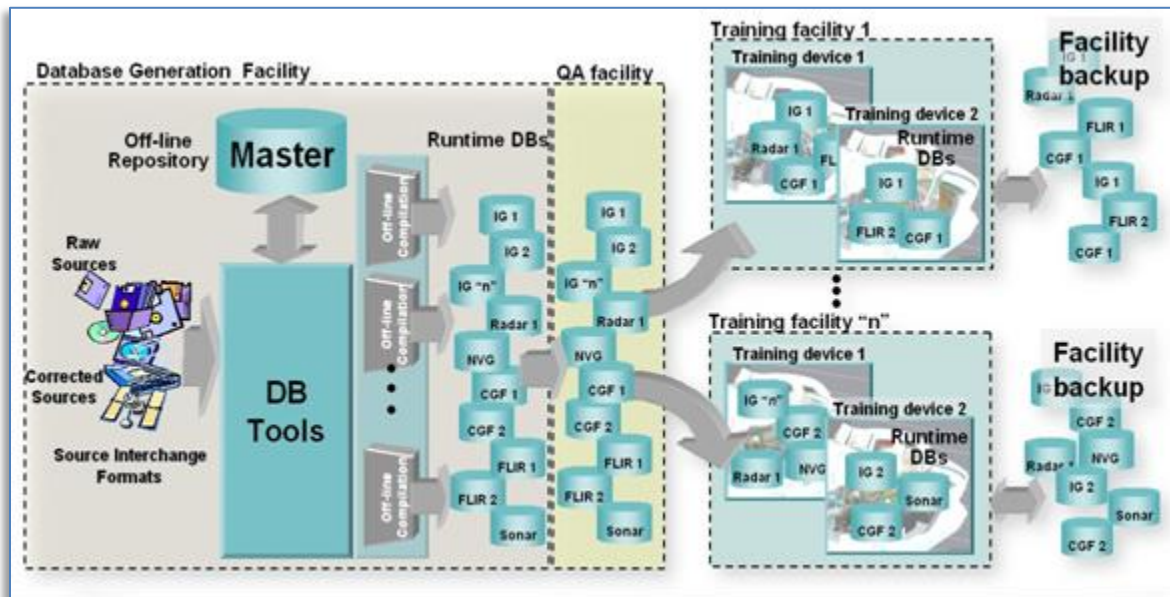


Figure 1-1: Use of CDB as an Off-line Database Repository

In most modern SE tool suites in-use today, the Data Preparation step shown in Figure 1-2: SE Workflow with CDB as an Off-line Database Repository consists of many sub-steps usually applied in sequence to each of the datasets (aka layers) of the SE. In effect, this aspect of the modeler's responsibilities is virtually identical to that of a GIS<sup>4</sup> specialist. As a result, many of the simulation equipment vendors offer SE authoring tools that integrate best-of-breed COTS<sup>5</sup> GIS tools into their respective tool suites. The steps include:

- **Format conversion:** raw source data is provided to modelers in literally hundreds of formats. Early on in the SE generation process, modelers typically settle on a single format per SE layer (e.g., terrain altimetry, imagery, attribution)
- **Error handling:** raw source often contains errors or anomalies that, if left undetected, corrupt and propagate through the entire SE data preparation pipeline.

<sup>4</sup> Geographic Information Systems

<sup>5</sup> Commercial-Off-The-Shelf



As a minimum, these errors must be detected early on in the process. More advanced tools can correct many of these automatically, particularly if there is some redundancy across the layers of data.

- **Data geo-referencing:** this is the process of assigning a unique location (latitude, longitude and elevation) to each piece of raw data entering the SE pipeline.
- **Data Registration:** each dataset is manipulated so that it coincides with information contained in the other datasets. These manipulations include projections, coordinate conversions, ortho-rectification, correction for lens distortions, etc. For images, this process is also known as rectification.
- **Data Harmonization:** the raw data of a dataset varies over a geographic extent if it was obtained under different conditions, such as from two or more sensors with differing spectral sensitivity characteristics, resolution, in different seasons, under different conditions of weather, illumination, vegetation and human activity. The modeler must factor for these variations when selecting and assembling the datasets into a self-coherent SE.

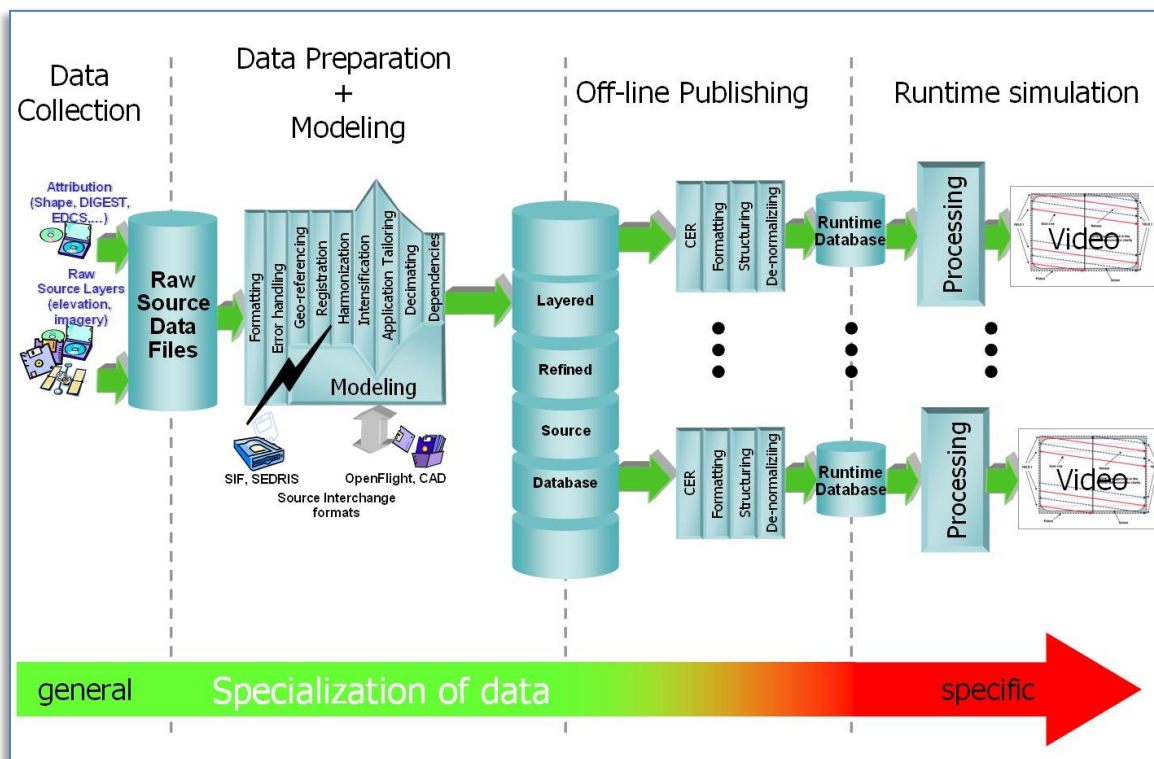


Figure 1-2: SE Workflow with CDB as an Off-line Database Repository

The effort expended during the Data Preparation and Modeling step is mostly independent of the targeted simulation devices and the targeted applications.

Consequently, the results of the data preparation step can be stored into a Refined Source DataBase (RSDB) and then re-targeted at modest cost to one or more simulation devices.

The standardization of RSDBs can greatly enhance their portability and re-usability. Unfortunately, standardization has not been the focus of most SE authoring tool vendors nor of simulation equipment vendors. The CDB now offers a standardized means to capture the effort expended during the Data Preparation and Modeling step. In effect, the CDB becomes a master repository where refined source can be “accumulated” and managed under configuration control.

While standardization of format/structure is essential to achieve high portability, inter-operability and re-use, the SE content must be ideally developed so that its content is truly independent of the training application. Therefore, we strongly recommend that the SE content of the CDB repository be developed to be independent of the training application.

Historically, SEs were developed for a single, targeted simulation application (e.g., tactical fighter, civil and air transport, rotary wing, or ground/urban warfare). In effect, the intended training application played an important role in determining the RSDB content because SE developers were constrained by the capabilities of the authoring tools and of the targeted simulation device. Unfortunately, this tailoring of SE was performed too early during the SE workflow and severely limited the applicability and re-use of the SE. Application tailoring can require either data intensification<sup>6</sup> or data decimation<sup>7</sup>. Note that the process of data intensification and decimation lends themselves to computer automation and can be performed during the off-line publishing process. In the future, we anticipate some data intensification to be done in real-time time.

Once the SE developer has completed his work in creating the various data layers of the Refined Source DataBase, he must off-line publish (aka “compile”) the SE into one or more device-specific data publishing steps. As we will discuss in section 1.3.1.2, Use of CDB as a Combined Off-line and Run-time Database Repository, the device-specific off-line compilation step can be entirely omitted if the targeted training equipment is CDB-compliant.

While an off-line publishing approach does not offer all of the benefits described in section 1.3.1.1, it nonetheless provides an easy, low-effort, migration path to CDB. Any equipment vendor can easily publish the RSDB into his proprietary runtime format. Firstly, the publishing process is facilitated by the fact that the CDB

---

<sup>6</sup> Data Intensification is the process of augmenting or deriving added detail from the information found in the raw data. For instance, intensification can be used to augment flattened terrain imagery with 3D cultural detail relief. A typical example of this consisting in populating forested areas found in the terrain imagery with individual three-dimensional trees.

<sup>7</sup> Data Decimation is the process of removing or simplifying the informational content found in the raw data. For instance, decimation can be used to transform individually modeled buildings into simplified city blocks or to reduce the resolution of terrain imagery. Data decimation is usually undertaken to ensure that the SE falls within the capabilities of the targeted simulator system.

specification internally use of industry standard formats (e.g., TIFF, JPEG, OpenFlight, and Shape). As a result, it bears much commonality with other database interchange formats. However, the CDB goes much further in that it “wraps” these formats into a global, standardized data model suited to high-end real-time simulations. This greatly facilitates the work of SE developers. Thus, the CDB provides a far simpler and straightforward means of interchanging refined source data, when compared to alternatives such as SIF, SEDRIS, OpenFlight that have partial pre-defined data models that are not sufficiently comprehensive for use in real-time simulations.

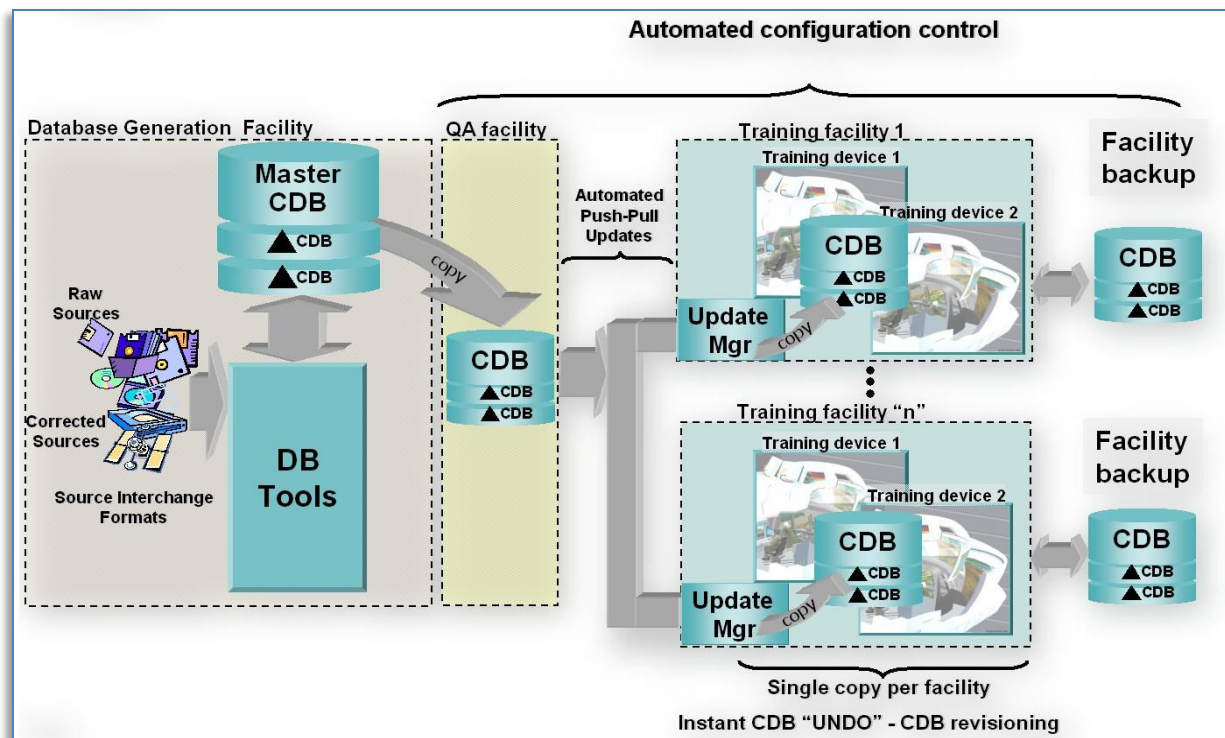
### **1.3.1.2 Use of CDB as a Combined Off-line and Run-time Database Repository**

A CDB can be both used an off-line repository for the database authoring tools or as an on-line (or runtime) repository for the simulators. When used as a runtime repository, the CDB offers plug-and-play interchangeability between simulators that conform to the CDB specification. Since the CDB can be used directly by some or all of the simulator client-devices, it is considered a run-time environment database.

In addition to the benefits outlined in section 1.3.1.1, Use of CDB as an Off-line Database Repository, the use of the CDB as a combined off-line and run-time repository offers many additional benefits, in particular:

- SE Plug-and-Play Portability and Interoperability across CDB-compliant simulators and simulator confederacies (be it tactical air, rotary, urban/ground, sea).
- Reduced Mission Rehearsal Timeline by eliminating SE generation steps (off-line publishing, database assembly and data automation)
- Simplified Deployment, Configuration Control and Management of Training Facility SE Assets by eliminating the duplication of SE runtime DBs for each simulator and each client-device of each simulator.
- Single, centralized storage system for the SE runtime repository (can be extended to a web-enabled CDB)
- Seamless integration of 3D models to the simulator.
- Fair Fight/Runtime Content Correlation through the adjustment of runtime level-of-detail control limits at each client-device.

Figure 1-3: Use of CDB as an Off-line and On-line Database Repository, illustrates the CDB as an off-line Master database repository for the tools and as an on-line Master database repository for the training facilities. Note that the deployment of the synthetic environment to the training facilities involves a simple copy operation. The deployment of the CDB is further simplified through an incremental versioning scheme. Since only the differences need be stored within the CDB, new versions can be generated and deployed efficiently.

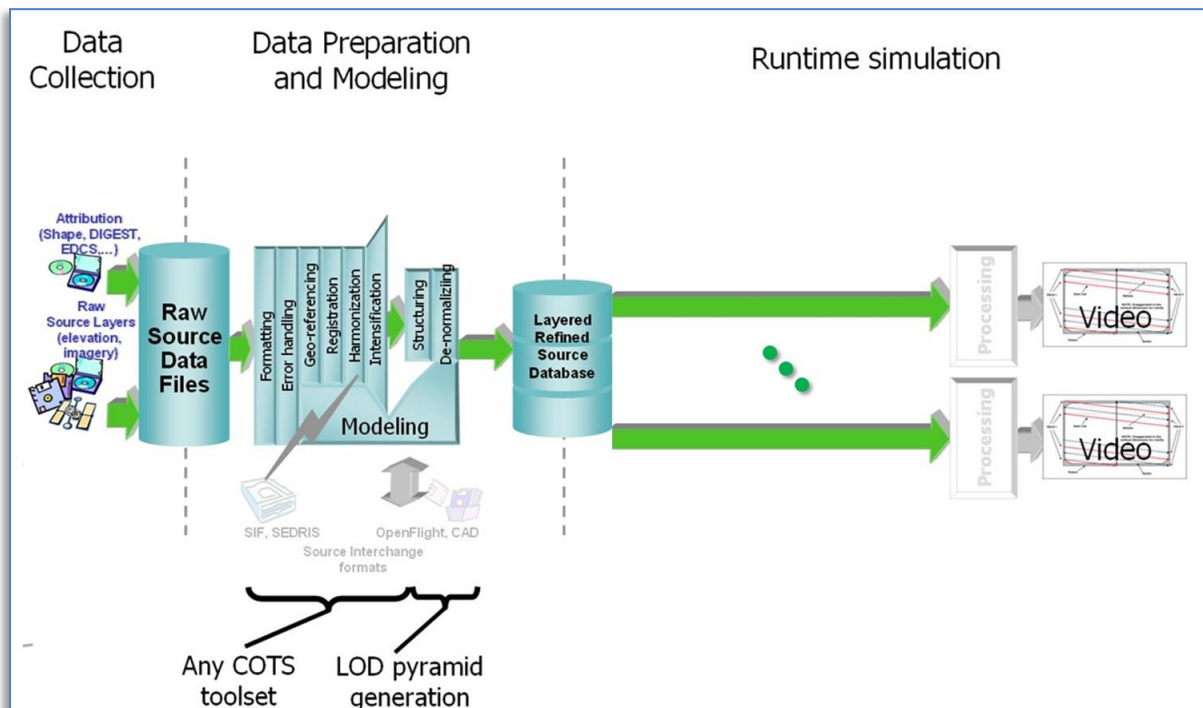


**Figure 1-3: Use of CDB as an Off-line and On-line Database Repository**

The CDB specification standardizes formats and conventions related to synthetic environments for use in simulation. However, many additional benefits can be garnered if the CDB is also used as an on-line database repository; this is particularly true when one considers the effort expended in the deployment of the synthetic environment to the training and/or mission rehearsal facilities.

When used as an on-line database repository, there is no need to store and maintain off-line published versions of the database for each client-device (as illustrated in Figure 1-3). As a result, the storage and computing demands on the database generation facilities are significantly lowered. This is especially true of database generation facilities whose mandate involves the generation of complex synthetic environments for use by several training facilities.

Figure 1-1: Use of CDB as an Off-line Database Repository, illustrates the simplified database generation workflow resulting from a CDB that is used as both an offline and a runtime SE repository.



**Figure 1-4: SE Workflow with CDB as Combined Off-line/Runtime Database Repository**

This approach permits the CDB representation of the synthetic environment to be “dissociated” from the resolution, fidelity, precision, structure and format imposed by the internals of client-devices. Compliancy to the CDB specification can be achieved either by modification of the client-device internal software to make it CDB-native or by inserting a runtime publishing process that transforms the CDB data into the client-device’s legacy native runtime format. In the later case, this process is done in real-time, on a demand-basis, as the simulator “flies” within the synthetic environment. Note that since the simulated ownship moves at speeds that are bounded by the capabilities of the simulated vehicle, it is not necessary to instantly publish the entire synthetic environment before undertaking a training exercise; the runtime publishers need only respond to the demands of the client-devices. When the simulated ownship’s position is static, runtime publishers go idle. As the ownship starts advancing, client-devices start demanding for new regions, and runtime publishers resume the publishing process. Publishing workload peaks at high-speed over highly resolved areas of the synthetic environment.

Note that virtually all simulation client-devices in existence today natively ingest proprietary native runtime formats. As a result, a runtime publisher is required to transform the CDB data into legacy client device’s native runtime format. The runtime publishing process is performed when the CDB is paged-in from the CDB storage device. Appendix A of the CDB Specification provides a set of guidelines regarding the implementation of Runtime Publishers.



### **1.3.2 What the CDB Specification Is Not**

The representation and sharing of synthetic environment 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. Their mandate was about:

1. providing a representation of synthetic environment data for the modeling and simulation field; and
2. providing the means to freely interchange synthetic environment datasets. Such Specifications addressed the interchange of data, representing the simulated natural environment, in advance of the runtime use.

The CDB Specification, on the other hand, is primarily intended as a distributed simulator runtime synthetic environment database Specification. It is a Specification optimized for distributed simulation applications that require very large synthetic environment databases for use in high-end simulation, mission planning/rehearsal. The CDB Specification datasets extend the simulation beyond just “classic out-the-window” visualization since it supports the simulation of sensors, such as FLIR, NVG, Radar and other simulation subsystems such as Computer Generated Forces and NAVAIDS.

The CDB Specification concerns itself with the modeled data representation and attribution of terrain, objects and entities within the synthetic environment. However, it does not concern itself with the movement, change in shape, physical properties and/or behavior of such objects and entities; this falls under the domain of synthetic environment simulations.

The CDB Specification does not concern itself with representations of celestial bodies (such as the Sun, Moon, stars, and planets). Rather, it assumes that the modeled representation (e.g., the Sun/Moon disk representation in an IG) of celestial bodies is internally held within the appropriate simulator client-devices.

Due to its real time vocation, the CDB Specification limits the number of units of measure for each physical quantity. For instance, all terrain surface coordinates are represented in lat-long while all other distances are in meters. This is in stark contrast to DB interchange standards, which permits a large set of units for each physical quantity (for example distance can be expressed in feet, meters, miles, parsecs, light-years, nautical miles, inverse-meters, angstroms, microns).

The CDB Specification provides a high degree of flexibility in setting content. There is no mandatory “coverage completeness requirement” imposed by the CDB Specification. This permits the generation of a CDB even when a database modeler is confronted with limited data availability. It is understood however, that a CDB that is minimally populated will be of limited value to many simulation applications. Hence, the applicability of any CDB is clearly related to the richness, quality and resolution

of its content. It is anticipated that additional standardization will be required to prescribe content appropriate to targeted simulation applications. In its current form, the CDB Specification does not mandate on synthetic environmental richness, quality and resolution.

Given the mandate of the CDB Specification to be platform independent, it cannot provide the implementation details of specific off-line database compilers or runtime publishers attached to specific client-devices. Furthermore, since there is no standardization of the SE representation internal to client-devices (they vary by type<sup>8</sup>, by vendors), it is unlikely that such information would completely satisfy the interests of all developers. More importantly, the structure and format of synthetic environment data ingested by each client-device is typically proprietary; as a result, it is impossible to fully describe the effort required to develop CDB off-line compilers and/or CDB runtime publishers.

### 1.3.3 What is a CDB

A CDB corresponds to a static synthetic representation of the whole earth; it is geographically divided into geodetic tiles (bound by latitudes and longitudes), each containing one or more specific sets of features. It uses the WGS-84 earth model to provide geodetic interoperability. Each of the simulator client-devices accesses the CDB geospecific information using the WGS-84 geodetic coordinate system.

A CDB contains the features and modeled representation of the synthetic environment. It contains terrain altimetry; its raster imagery, its attribution as well as 3D feature with their modeled geometry, texture and attribution. The Specification also makes provision for the representation of moving models. The representation of moving models is comprehensive and goes well beyond other visualization standards because it makes provisions for the standardized simulator naming conventions, material and feature attribution, radar/laser reflectivity, aircraft and airport lighting systems, armaments, special effects, collision points/volumes just to name a few.

The Specification governs all aspects of the CDB, as follows:

- Data content and representation of the synthetic environment
- Structure and organization of the synthetic environment
- File format of the synthetic environment as stored on media

The CDB Specification describes a modeled synthetic environment representation for distributed simulation applications. Section 1.6.5, Use of CDB in Applications Requiring Dynamic Synthetic Environments discusses how a CDB-compliant simulator could be adapted to handle modifications of the synthetic environment in real-time.

Given the CDB's structured representation and attribution of terrain, objects and entities, it is now possible to design a broad range of synthetic environment

---

<sup>8</sup> It is precisely the intent of the CDB Specification to provide such standardization.



simulations that modify synthetic environments in real-time. Such simulations can modify the CDB and notify their changes to the other simulation federate that share a CDB. This provides a synthetic environment that is persistent and fully correlated across all simulation federates. For example, terrain trafficability could be handled by a new SE simulation that dynamically calculates soil moisture content as a function of localized rain precipitation and soil types/composition. A second simulation would then derive the resulting soil physics and determine vehicle wheel slippage across the varying terrain conditions.

The modification/notification approach is well-suited for a broad gamut of SE simulations; however, some simulations are very data intensive and would require excessive broadcasting bandwidths to other federates. In such cases, these dynamic simulations would have to be replicated in the other client-devices of the federation. Good examples of this are visual system special effects (e.g., rotor downwash effect, missile plumes, muzzle flashes, cast landing lights); typically such simulations are proprietary and intrinsic to the client-devices. Other examples of this include the varying effects of weather<sup>9</sup> (local winds, temperature, humidity, particulates, etc.) and oceans (currents, temperature, etc.).

## **1.4 Key Features and Characteristics of the CDB Specification**

The following paragraphs provide an overview of key features and characteristics of the CDB Specification.

### **1.4.1 Synthetic environment Database for Simulation Applications**

The CDB Specification is a simulator-level synthetic environment database Specification for use in real-time. The Specification is open, platform-independent and client-device independent. The Specification defines all aspects of data representation and organization, and storage structure necessary to support full-mission real-time simulation. The CDB Specification synthetic environment databases contain datasets that represent the features of a synthetic environment for the purposes of simulation applications. The CDB provides a time-invariant synthetic environment representation of the Earth, composed as terrain, natural/man-made features and moving models for use by SE authoring tools and simulators.

### **1.4.2 Logical Addressability**

The CDB Specification provides for near-infinite addressability. The amount of information associated with a CDB is limited only by available disk storage. The CDB Specification allows for a nearly infinite number of identifiers within a single CDB. Addressability is not expected to be a limiting factor even if we assume a yearly doubling of storage capacity.

---

<sup>9</sup> Time-varying weather simulation effects could be simulated by a “weather server” simulation subsystem which in turn can rely on the terrain elevation and terrain material datasets to perform its simulation of weather in real-time.

### **1.4.3 High Spatial Resolution and Scalability**

The CDB Specification provides for near-infinite resolution due its organization of data in LODs. The CDB Specification grid-organized data (e.g., elevation, ground raster imagery, ground properties) is structured into an LOD hierarchy consisting of up to 34 LODs. Grid-organized data can be represented to a resolution of 13 microns.

The CDB Specification also provides for near-infinite cultural content. The CDB Specification for vector data (3D point, lineal and areal features) is also structured into a LOD hierarchy consisting of up to 34 LODs. At the finest possible LOD, a CDB can contain in excess of 100 million cultural features per square meter.

The CDB Specification permits each geographic area to be modeled at a distinct LOD in accordance to the availability of source data. Since this capability is provided at a tile-level, it is storage-efficient.

### **1.4.4 Earth Geodetic Spatial Representation Model**

Geo-referenced data constitute a major aspect of the CDB Specification synthetic environment data. It uses a geographic coordinate representation (WGS-84 lat/long, elevation) for the earth's terrain surfaces and ocean floor. Furthermore, natural and man-made objects are positioned and oriented using geodetic coordinates. The CDB Specification provides geodetic coverage for the entire earth.

### **1.4.5 Tile/Layer/Level-of-Detail Structure**

The CDB Specification offers a structure that is well suited for the efficient access of its contents. To this end, it relies on three important means to organize the synthetic environment data:

- Tiles
- Layers
- Level-of-Detail (LOD)

#### **1.4.5.1 Tiles**

The CDB Specification relies on a top-level tile structure designed to organize the data for efficient access in real-time. The tile structure provides an effective means for simulator client-devices to access the datasets of a geographical area at a selected LOD. Since the spatial dimension of tiles varies inversely with LOD (i.e., resolution), client-devices can readily predict the amount of data contained within the tile; as a result, the management of memory within client-devices is simplified and much more deterministic.

The CDB Specification Data Representation Model (DRM) is logically organized as a strip of geo-unit aligned tiles along each earth slice. Each earth slice is divided into a decreasing number of tiles to account for the fact that the length of an earth slice decreases with increasing latitude.

### **1.4.5.2 Layers**

The CDB Specification DRM is also logically organized as distinct layers of information. The layers are theoretically independent from each other, (i.e., changes in one layer do not impose changes in other layers).

Layers are additive in fidelity, (i.e., the achievable level of the simulation fidelity increases with the number of layers).

Secondly, unavailable layers are automatically defaulted. A database modeler need not fully populate the CDB. There is no mandatory “coverage completeness requirement” imposed by the CDB Specification. This feature permits the generation of a CDB even when database modelers are confronted with limited data availability. The usefulness/faithfulness of the synthetic environment increases with the number of available layers.

The layering mechanism improves the efficiency of the client-devices since they need only access (and be aware) of the datasets that are relevant to them, at their prescribed level of fidelity. For instance, a simulator CGF client-device will not likely have a reason to use ground raster imagery; however, it is quite likely interested in accessing ground surface properties when determining traffic-ability.

### **1.4.5.3 Levels-of-Detail**

The availability of LOD representations is critical to real-time performance, especially when dealing with grid-organized data. 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 ownship. As a result, many client-devices can reduce (by 100-fold or more) the required bandwidth to access synthetic environment data in real-time.

Additionally, the availability of LODs can be readily exploited by simulator client-devices to improve their algorithmic efficiency. Devices such as Image Generators (IGs) can readily take advantage of an LOD structure because image perspective naturally reduces image detail with distance as a result of the perspective computation inherent to the IG; as a result, much less geometric detail and texture detail need be processed or considered at far range.

The spatial sampling pitch of each successive LOD follows power of two progressions; this is a pre-requisite for the efficient and deterministic management of memory by all client-devices of a typical simulator.

The terrain LOD can be controlled to the tile level. Since the CDB Specification supports a variable LOD hierarchy spanning up to 34 levels, it is possible to control the allocation of LODs by area. The variable LOD hierarchy provides for efficient use of storage because the LOD hierarchy needs to be deepened only in the areas where higher resolution is desired. Figure 1-5: Variable Allocation of LOD, illustrates an earth strip made up of a series of tiles; some portions of the strip have been modeled with 3, 4, or 5 LODs according to the application requirements.

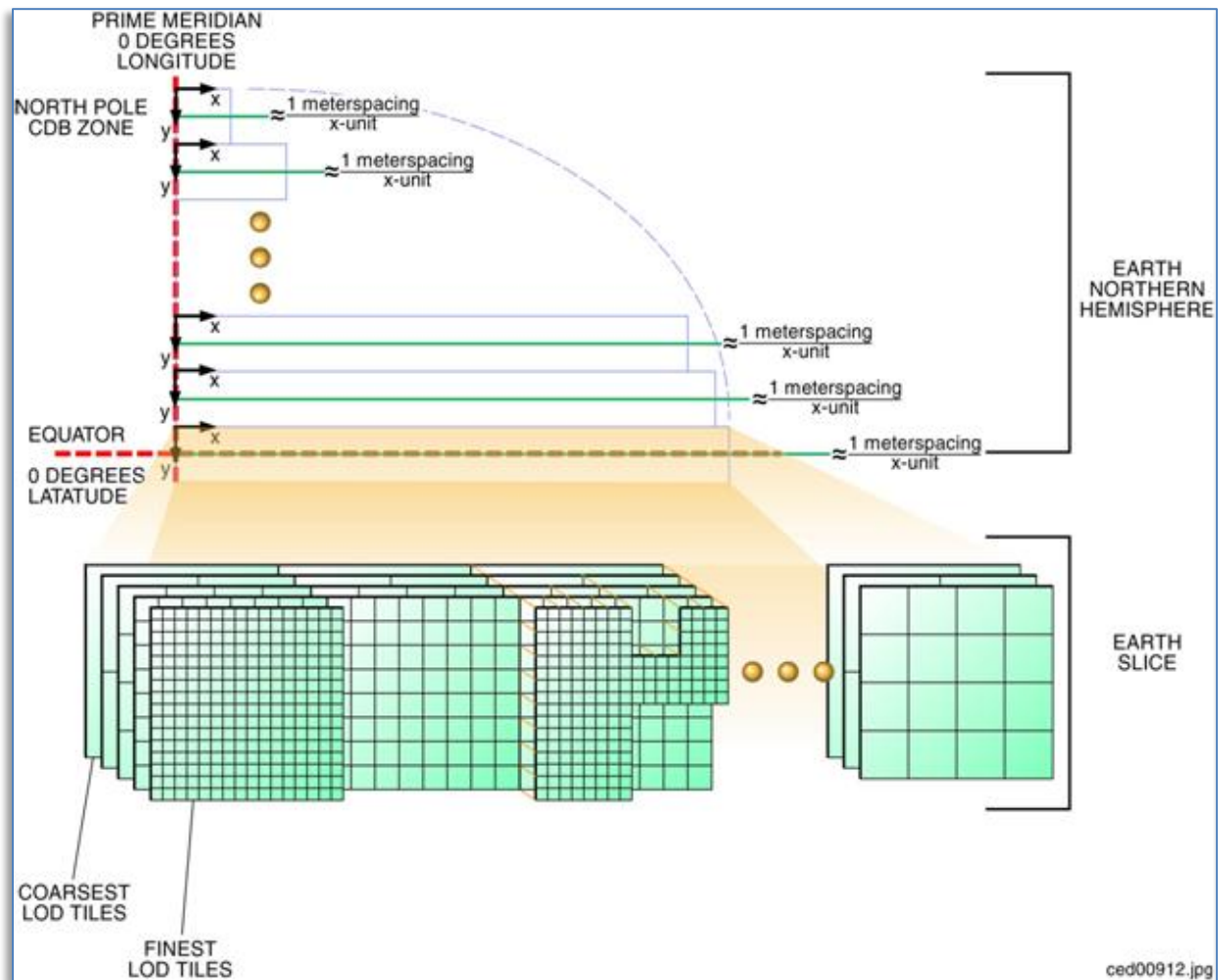
#### **1.4.6 Platform Independence**

The constraints imposed by the CDB Specification are minimal and are designed to allow its implementation on any of the widely available computer hardware platforms, operating systems, file systems and transport protocols.

On CDB-compliant simulator client-devices<sup>10</sup>, any CDB can run “as-is” without the traditional off-line compilation step. This allows the simulation user community to freely exchange CDBs across simulators either through the exchange of physical media (or entire storage subsystems) or via network. As a result, a CDB can run without change regardless of the computer platforms, simulator system software and simulator client-devices. While the CDB Specification does not explicitly mandate the use of specific computer platforms and system software, it does however specify a set of minimum criteria as listed below.

---

<sup>10</sup> A CDB-compliant client-device is a client-device that inputs synthetic environment data that conforms to the format, structure and conventions of this Specification. Any client-device that does not natively conform to this Specification must reformat and restructure the CDB data to the device’s native format/structure through the use of a runtime publisher. A client-device need not input all of the CDB datasets and attributes to be CDB-compliant.



**Figure 1-5: Variable Allocation of LOD**

### 1.4.6.1 System Software Independence

The following paragraphs explain the various software requirements.

#### 1.4.6.1.1 File System

The CDB Specification is file system independent, (i.e., it does not mandate the use of a specific file system). However, compliance to the CDB Specification does require that the file system be able to support a minimal set of capabilities. All modern file systems in use today conform to the minimal set of capabilities described in Section 2.2, File System File Naming Conventions.

#### 1.4.6.1.2 Operating System

The CDB Specification is Operating System (OS) independent; it does not mandate the use of a specific OS. However, compliance to the Specification does require that

the operating system be able to support a minimal set of capabilities. All modern operating systems in use today conform to the minimal set of capabilities listed below. This includes Windows32, NT, UNIX, Linux, IRIX, Solaris, etc.

The CDB Specification assumes that the operating system supports, as a minimum, the following:

1. Byte-stream random file access
2. 32-bit integers, natively
3. A 32-bit address space
4. Floating point support (per IEEE-754), natively
5. 2GB virtual address space per process
6. Memory paging
7. Network communication

#### **1.4.6.1.3 Transport Protocols**

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

#### **1.4.6.2 System Hardware Independence**

The CDB Specification is hardware independent; it does not mandate the use of particular hardware platforms. Furthermore, any general-purpose hardware compatible with modern Operating Systems (OS) can be used for a CDB Specification implementation<sup>11</sup>.

The CDB Specification assumes that the system hardware supports, as a minimum, the subsystems described in the following sections.

##### **1.4.6.2.1 Processor/Memory**

The CDB Specification is processor-independent; it does not mandate the use of a specific processor type. However, compliance to the CDB Specification does require that the processor be able to support a minimal set of capabilities. All modern processor systems in use today conform to the stated minimal set of capabilities listed below. This includes processors such as the Pentium, XEON, Celeron, Duron, Athlon, MIPS, and PowerPC. The CDB Specification does not impose requirements on hardware/memory over and above those mandated by the commonly available OS; as a minimum, the CDB Specification itself requires support for:

1. 8-bit, 16-bit, and 32-bit signed and unsigned integers, natively

---

<sup>11</sup> The CDB Specification internally uses conventional “atomic” data types, namely signed/unsigned  $n$ -byte quantities, ASCII strings, IEEE-754 floats (single/double). While it is theoretically possible to build a CDB Specification-compliant software implementation on a processor platform that does not explicitly support these atomic types, it is clear that such implementation would be inefficient.



2. A 32-bit address space
3. 32-bit and 64-bit double precision floating point values (IEEE-754), natively
4. 2 GB virtual address space
5. Virtual memory space
6. Immediate and indirect memory addressing modes

#### **1.4.6.2.2 Storage Subsystem**

The CDB Specification does not impose requirements on the storage subsystem other than those mandated by the selected OS and the selected file system. As a minimum, it must be able to support file system as previously specified. An empty CDB requires very little in terms of storage; it will likely fit on a few diskettes or a single CD. However, a typical CDB is expected to range from hundreds of Mbytes to hundreds of Tbytes.

#### **1.4.6.2.3 IO Subsystem**

The CDB Specification does not impose requirements on its IO subsystem other than those mandated by the selected OS, specifically:

1. Disk I/O subsystem
2. Network I/O subsystems

#### **1.4.6.2.4 Client-Device Independence**

The CDB Specification is client-device independent; it caters to the collective needs of all client-devices likely encountered on a tactical mission simulator; the Specification itself is completely independent of any vendor-specific devices.

**NOTE:** Each client-device is matched either to an off-line compiler or to a runtime publisher. In the runtime case, the runtime publisher transforms this data into the client-device's legacy native data format and structures the CDB synthetic environment data as it is paged-in by its client-device. Regardless of its use as an off-line or on-line repository, the CDB eliminates all client-format dependencies. Alternately, the client-device may be designed / modified to be CDB-native, in which case a separate runtime publisher is not required. Note that the CDB Specification makes use of data types commonly available in standard computer platforms (floats, integers, etc.). While it would be theoretically possible to cater to a client-device that does not support the CDB Specification "atomic" data types, it would unduly load the attached on-line publisher. As a result, it is recommended that all client-devices provide hardware support for the CDB atomic data types.

The CDB Specification limits its scope to synthetic representations of the earth; its internal data is organized and optimized to reflect its intended use in simulation applications. The concept of an earth model and the means to represent earth surface



regions with associated cultural features figure prominently within the Specification because they are important to the targeted CDB applications<sup>12</sup>.

Since it is the client-devices that initiate access to the CDB, they must each be theoretically “aware” of at least the geodetic earth model. Otherwise, the contents and the structure of the CDB can be completely abstracted from the client-device. The runtime publishers provide the necessary level of abstraction. The level of abstraction provided by the publishers is purely a function of the selected implementation of the client-device and its associated publishers. It is entirely acceptable, for the client-device to understand and to be completely “aware” of the CDB as it is defined by the CDB Specification. In this case, such a device would not require a runtime publisher, because it is already CDB-native.

The runtime publishers bridge the gap between the information content requested by the attached client-device and the information content and structure available to them in the CDB. As a result, the runtime publishers must each be theoretically “aware” of the following CDB concepts:

1. Tile:  
Ability to understand the concept of earth surface areas hierarchically subdivided in accordance to the CDB Specification
2. Level of detail:  
Ability to understand the concept of resolution or level-of-detail, for terrain, cultural vector data, raster imagery, and model geometry
3. Terrain surface representation:  
Ability to understand concepts pertinent to earth surface geometry and attribution
4. Cultural vector data (point, linear, areal):  
Ability to understand the concept of point, linear and areal cultural data and related attribution, fixed or conformed to earth surface
5. Grid-organized data and meshes:  
Ability to understand the concept of grid-organized single-value datasets (e.g., elevation grid) and multi-value datasets (e.g., color triplets)
6. Imagery:  
Ability to understand the concept color raster imagery mapped onto terrain surfaces or models
7. Model geometry (incl. light points):  
Ability to understand the concept of general surface geometry

---

<sup>12</sup> Conversely, the Specification is not optimized to represent CAD/CAE data for use in manufacturing.

8. Model positioning capability:  
Ability to differentiate between statically and dynamically positioned models
9. Descriptive attribution:  
Ability to understand the attribution concepts pertinent to the client-device

#### **1.4.7 Synthetic Environment Scalability & Adaptability**

The concept of scalability when applied to synthetic environments not only applies to the geographic extent of the database but more importantly, it also reflects the ability to scale the environment in resolution and fidelity. These concepts are fully supported by the CDB Specification and are explained below:

1. Geographic extent:  
Correspond to the range of geographic extent of the earth surface that can be modeled.
2. Resolution:  
Correspond to the range of information density (for instance, the number of elevation values per km<sup>2</sup>) of the modeled datasets.
3. Fidelity:  
Correspond to the amount and type of synthetic environment data that can be modeled to support higher-fidelity simulator client-devices<sup>13</sup>. Consider for instance a simulator capable of supporting a single-surface earth skin representation versus one capable of representing tunnels, bathymetric data, location-dependent tide heights, location-dependent wave heights, etc. Clearly, the amount of information required by the higher-fidelity simulator is greater.
4. Precision:  
Correspond to the range of numerical precision (i.e., number of bits allocated to represent a quantity) of the modeled datasets.

Today, modelers face difficult challenges if they want a synthetic environment that is both scalable and adaptive. The solution to this difficult issue extends beyond the “mechanics” of achieving backward/forward compatibility; it also requires a complete “dissociation” of the database from any of the constraints imposed by the client-devices. It is current practice today for modelers to repeatedly adjust the

---

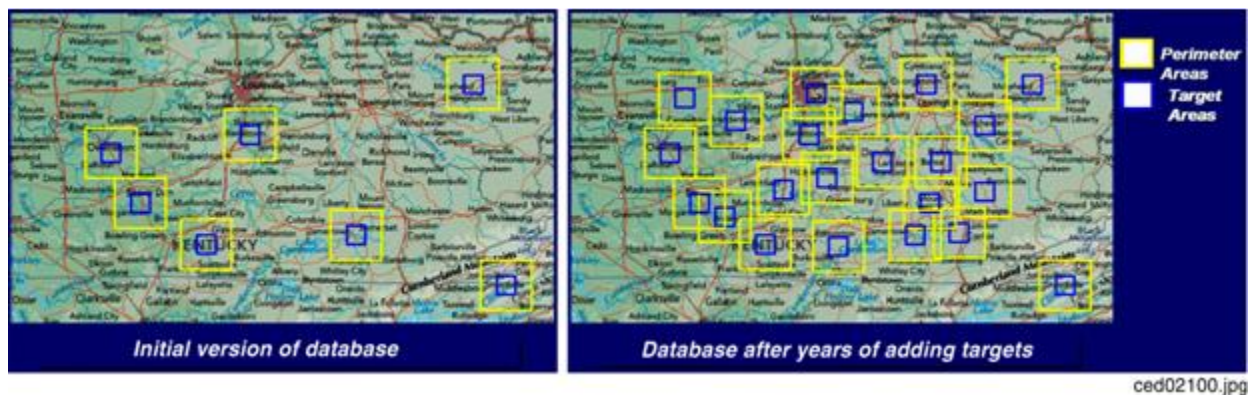
<sup>13</sup> The CDB Specification supports this concept through two mechanisms:

Data defaulting: The CDB Specification is tolerant to missing data, because all attributes, layers and datasets have Specification default behaviors. As more of the data is provided, the fidelity of the environmental database increases.

Additive layering: The CDB Specification offers a layered environment database organization. The layer organization is “fidelity-ordered”, (i.e., basic layers appear first in the hierarchy while the layers needed for a higher-fidelity simulation appear later in the hierarchy).

content, resolution and density of synthetic environment databases to closely match the capabilities and performance of the targeted client-devices. When content is added to the database, previously modeled content is usually removed or simplified. Under such constraints, it is difficult for a modeler to build a synthetic environment database capable of meeting anything but its immediate requirements.

Figure 1-6: Typical Evolution of a Database shows the typical evolution of a modeled region for use in a mission rehearsal. The initial version of the region may have only a few high-resolution/high-fidelity areas; however, over a given time period, modelers will be asked to model additional target areas. As a result, the size, complexity, resolution and fidelity of the synthetic environment database gradually increase. Without built-in provisions for scalability, significant rework of the database is required each time a target area is added.



**Figure 1-6: Typical Evolution of a Database**

The CDB Specification, on the other hand, offers a new paradigm: the “dissociation” of the synthetic environment database’s extent, resolution, fidelity, precision, structure and format from its client-devices. This concept is not limited to aspects of formatting, numerical representation, internal structure, file structure, file systems, etc. More importantly, the concept also covers aspects relating to synthetic environment database fidelity and resolution. Historically, the fidelity and resolution of synthetic environment databases has been intimately linked to the capabilities of the targeted simulator client-devices. More often than not, the source data was manipulated and adapted to constraints imposed by the client-devices. As a result, the content, resolution and density of synthetic environment databases were repeatedly adjusted to closely match the capabilities and performance of the targeted client-devices. The amount of time devoted to repeatedly adding/editing/removing content, and then repeatedly re-publishing would largely exceed the effort of creating and building the original synthetic environment database. The CDB Specification offers the means to break this inter-dependence.

When assembling a CDB, the synthetic environment database modeler is permitted (within their time-cost budget) to include content that significantly exceeds the capabilities of their simulator(s). The job of “adjusting” content to client-devices is

relegated to the runtime publishers; the modeler is freed from this labor-intensive, repetitive task.

In a typical CDB Specification implementation, it is anticipated that client-devices will independently control their respective runtime publishers so that the runtime published synthetic environment corresponds to their inherent capabilities (resolution, fidelity, etc.).

Nonetheless, the runtime publishing paradigm offers interesting new possibilities. For instance, it would be possible to individually adjust the fidelity and resolution of the synthetic environment for each client-device; this adjustment could be done at any time. As a result, it is possible to control and adjust the overall simulator fidelity and correlation to a level that was previously unachievable.

### **1.4.8 Platform Scalability**

The CDB Specification does not enforce a particular computer hardware infrastructure. The selected infrastructure allocated to the handling of the CDB is largely determined by the overall simulation requirements. Any leeway the simulator architect may have at their disposal when trading-off various simulator performance parameters against each other, are as follows:

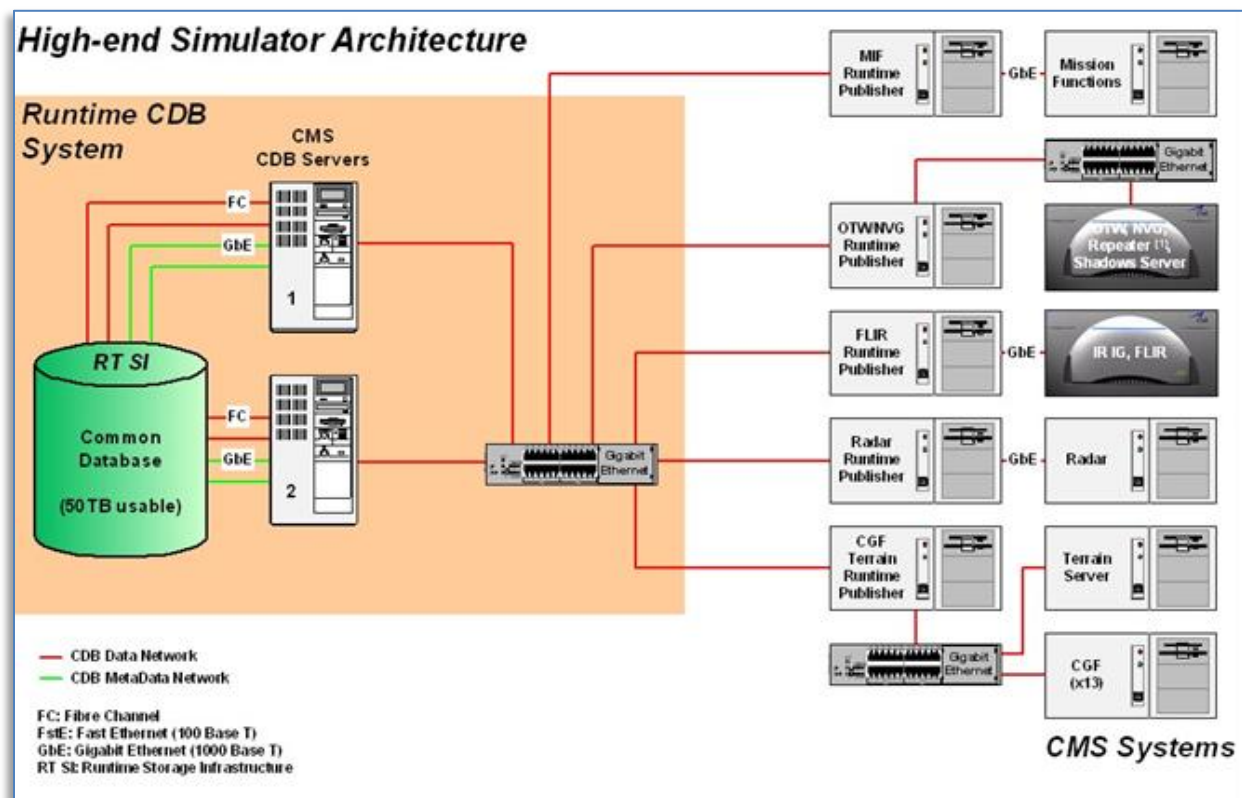
1. Geographic extent
2. SE/Simulator Resolution
3. SE/Simulator Fidelity
4. SE/Simulator Precision
5. Desired level of client-devices correlation
6. Client-level SE load management
7. Simulated aircraft speed
8. CDB sharing

An experienced simulation engineer can typically undertake this analysis; however, the process requires a good understanding of the content available in the targeted CDB(s) and of the content each client-device needs in order to meet its stated level of performance and fidelity. If for cost considerations the system engineer is unable to reconcile both, he can relax or trade-off some of the above parameters.

Alternately, it is possible to implement a simulator with client-devices (or attached publishers) that are capable of automatically adjusting resolution and fidelity in order to overcome performance limitations attributable to the CDB storage infrastructure and/or runtime publishers.

Figure 1-7: Typical Implementation of CDB Specification on High-end Simulator, below is a system block diagram of a typical implementation of the CDB Specification on a high-end tactical flight simulator. The runtime CDB system serves the combined needs of a mission functions computer, an OTW/NVG IG, a FLIR IG, Radar and a CGF subsystem equipped with its own terrain server. The runtime CDB system is scaled to reflect the collective capabilities of the attached client-devices; as a result, the storage system is configured to supply the necessary bandwidth.

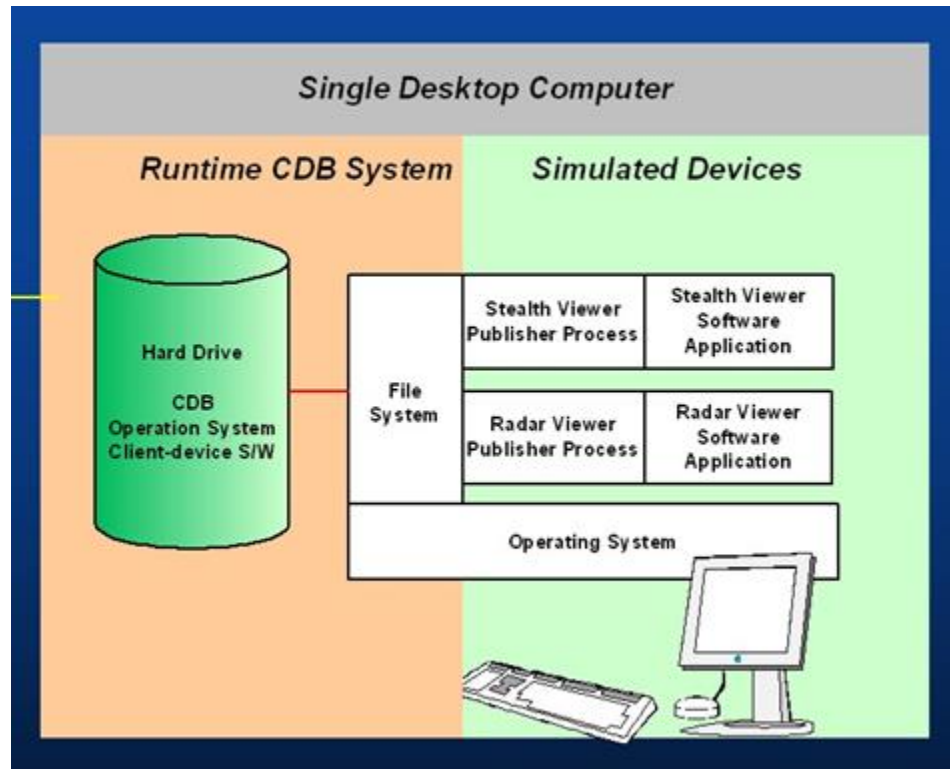
Similarly, the IO bandwidth of the CDB servers and processing performance of the runtime publishers are scaled to satisfy the demands of their respective client-devices.



**Figure 1-7: Typical Implementation of CDB Specification on High-end Simulator**

Figure 1-8: Typical Implementation of CDB Specification on Desktop Computer, below shows a modest application of the CDB Specification; it consists of a single desktop computer equipped with both stealth viewer and radar simulation application software. Each application is front-ended by a runtime publisher that in turn interfaces to the CDB via a standard file system. Given the more modest platform resources, some trade-offs in either resolution or fidelity might be required. This can be implemented in the runtime publisher or in the client-device application software.





**Figure 1-8: Typical Implementation of CDB Specification on Desktop Computer**

#### 1.4.9 Simulator Wide Unique Data Representation, Data Normalization

A CDB is a single repository for the entire simulator's synthetic environment DB. The CDB's internal structure ensures that all datasets are uniquely represented yet available (through a publishing process) to each of the simulator client-devices at runtime. This paradigm eliminates the extensive duplication of datasets that are shared by two or more client-devices.

The CDB Specification is technically a normalized data model in the sense that it best meets the logical data design objectives of correctness, consistency, simplicity, non-redundancy and stability. Ignoring any tailoring or tuning for particular application requirements or performance, a normalized design provides the following advantages:

1. Minimize amount of space required to store data:  
Normalization precludes storing data items in multiple places.
2. Minimize risk of data inconsistencies within the database:  
Since datasets are stored in only one place, there is no risk of datasets becoming inconsistent (uncorrelated).
3. Minimize possible update and delete anomalies:  
A normalized CDB eliminates the concerns related to update or delete operations.

4. Maximize the stability of the data structure:  
The process of normalization helps in associating attributes with entities based on the inherent properties of the data rather than on particular application requirements. Thus, new application requirements are less likely to force changes on the CDB Specification database design.

#### **1.4.10 Compression of Storage Intensive Imagery Datasets**

The CDB Specification prescribes the use of an industry standard compression algorithm for its storage intensive raster imagery datasets. This not only provides a substantial reduction in storage, but also reduces the data transmission bandwidths associated with simulator's access to the synthetic environment database at runtime. As a result of its storage efficiency, the CDB Specification relies on relatively few data formats for storing its datasets; there is no benefit (other than storage efficiency) to be gained in supporting any other specialized data formats whose underlying objective is only for storage efficiency. The CDB Specification embodies the JPEG 2000 industry standard format for raster imagery because it has comparable storage efficiency to all of these image formats without sacrificing any generality. JPEG 2000 has been chosen by the CDB Specification as a format for the storage of OTW raster imagery because of the following characteristics:

1. High compression efficiency:  
Compression better than 0.25 bits per pixels. Virtually indiscernible loss in image quality for 10:1 – 20:1 compression.
2. Lossless and lossy compression:  
Lossless compression ratios approx. 1.7:1
3. Perceptual color space internal coding:  
Allow dark images to be reconstructed without banding artifacts.
4. High dynamic range:  
Compress and decompress images with various dynamic ranges (e.g., 1-bit to 16-bit) for each color component.
5. Large images sizes:  
Up to  $(2^{32} - 1)$

There are other characteristics of the JPEG 2000 that are worth mentioning but are not directly beneficial to the CDB Specification. Those are:

1. Progressive image reconstruction:  
Allow images to be reconstructed with increasing pixel accuracy and resolution.
2. Region of interest coding:  
Permits certain Region of Interest (ROI's) in the image to be coded and transmitted with better quality and less distortion than the rest of the image.



3. Seamless quality and resolution scalability:  
Without having to download the entire file
4. Error resilience during transfers.

JPEG 2000 will be solely targeted at Raster Imagery data only. The reason is simply because of its highly efficient compression scheme that fits well with the goal of reducing the huge datasets associated with Imagery. Other raster-based datasets defined in the CDB will solely be using the TIFF format due to their more manageable size.

#### **1.4.11 Compression of other Raster Datasets**

In general, all TIFF/GeoTIFF files benefit from LZW compression when their content is not of type floating-point. For this reason, and as a general practice, the CDB Specification recommends the compression of all TIFF-based raster datasets containing integer values.

### **1.5 Key Benefits of the CDB Specification**

The following outline the key benefits of using the CDB Specification.

#### **1.5.1 Improved Synthetic environment DB Generation Timeline**

Important reductions in both the DB generation and DB update timeline are expected from the CDB Specification because:

1. There is no need to compile distinct synthetic environment runtime databases for each of the simulator client-devices.
2. All of the datasets common to two or more client-devices need not be duplicated. All associated client-specific structures are also eliminated.
3. Tiles and layers are technically independent from each other; as a result, there is no need to reprocess neighboring tiles and coincident layers. However, one exception to this relates to the level-of-detail generation.
4. The CDB Specification tile structure permits users to “pipeline” or overlap the DB creation/update process, see Figure 1-9: Pipelined DB Update Process, with DB transfer process<sup>14</sup>.

---

<sup>14</sup> Assuming the DB toolset (used by the modelers at the DB Generation Facility) allow its users to manipulate Environmental DB content on a small-area basis.

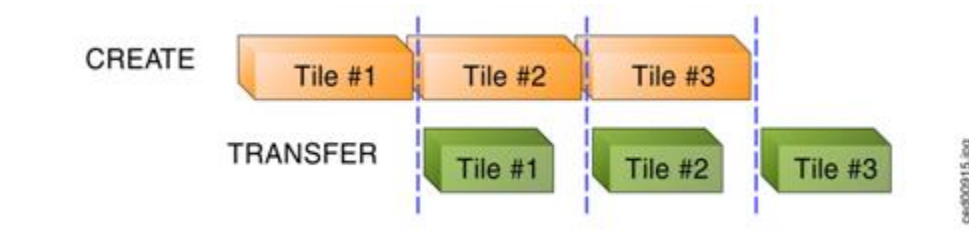


Figure 1-9: Pipelined DB Update Process

5. The CDB Specification tile structure lends itself naturally to the concurrent or “parallel” production of the CDB.
6. The CDB Specification internal versioning mechanism lends itself well to CDB updates because only the affected tiles or layers need be re-transmitted to the simulator. This represents a significant timesaving, especially in cases where small updates are constantly applied to a comparatively large CDB.
7. The conversion of the synthetic environment from its tool-native representation into a form directly entered by each of the simulator client-devices is broken down into several publishings accomplished in real-time on behalf of each of client-device. This approach permits the CDB Specification representation of the synthetic environment to be “dissociated” from the resolution, fidelity, precision, structure and format imposed by each client-devices. Collectively, the runtime publishers handle the more complex portion of the workload; they transform the CDB data into a form that is ingested on-the-fly by the client-device. This process is done in real-time, on a demand-basis, as the simulator progresses within the synthetic environment. Since there is one publisher for each of the client-device, the publishing workload is typically distributed across several computer platforms. Furthermore, since the simulated ownship moves at speeds that are bounded by the capabilities of the simulated vehicle, it is not necessary to instantly publish the entire synthetic environment before undertaking a training exercise; the runtime publishers need only respond to the demands of the client-devices. When the simulated ownship’s position is static, runtime publishers go idle. As the ownship starts advancing, client-devices start demanding for new regions, and publishers resume the publishing process. Publishing workload peaks at high-speed over highly resolved areas of the synthetic environment.

### 1.5.2 Interoperable Simulation-Ready Synthetic environment DB

A CDB-compliant database CDB is a fully interoperable, simulator-ready synthetic environment DB, (i.e., it can be used “as-is” on any CDB-compliant simulator). Because the CDB is free of any simulator dependencies, there is no need to undertake a time-consuming and expensive rework of the runtime database(s) in order to adapt it (them) to the format, structure, content, and precision constraints imposed by the simulator.



### **1.5.3 Improved Client-device Robustness/Determinism**

The CDB Specification tile organization provides the means to implement deterministic loading and/or paging of the CDB because each tile corresponds to the same amount of data (i.e., a “quanta” of information called a LOD-tile), regardless of its position on earth and regardless of its assigned LOD. This is a key characteristic of the CDB Specification and is necessary to ensure deterministic operation of client-devices, even when the CDB’s resolution varies considerably from region to region or when the CDB is modeled at an extremely high-resolution. It is quite straightforward for a client-device to determine the amount of memory it must locally allocate when loading (or paging-in) a geographical area of interest. If the geographical area of interest exceeds the client-device’s capabilities, it can easily revert to a coarser LOD, hence gracefully degrading, as opposed to aborting (due to an internal failure in allocating sufficient memory) or “skipping” over the offending area.

Consider for instance, a CDB modeled to a background area terrain texture resolution of 10 m, and a target area modeled to a resolution of 1cm. The target area holds 1,000,000 fold more data per unit of geographic area than the background area. Furthermore, assume that the client-device is not designed to take advantage of the CDB LOD tile structure, i.e., it always loads or pages its data as fixed-dimension (geographically speaking) areas. Upon requesting a fixed-dimension portion within the target area, the client-device would receive 1,000,000 fold more data than in the background area. Clearly, such a device would require an enormous amount of physical memory to alleviate such a variation in resolution. The CDB Specification offers a more manageable alternative because the CDB is allocated into fixed-sized LOD tiles. Hence, the client-device’s paging manager can first break down the requested area into tiles at the requested LOD; from this, the paging manager can easily predict the amount of storage needed to hold the requested area in memory; if it exceeds the client-device capabilities the client-device can revert to a coarser LOD.

The CDB tile organization lends itself to robust, deterministic management of memory within client-devices because memory can be allocated/de-allocated in fixed sized quantities. As a result, client-devices need not concern themselves with complex and non-deterministic memory de-fragmentation schemes that do not lend themselves to real-time applications.

### **1.5.4 Runtime-Adjustable Synthetic Environment DB Correlation and Fidelity**

The CDB Specification plays a critical role in improving the internal correlation of a synthetic environment because it eliminates the replication of runtime databases for each of the client-devices. The prior art in simulation mandated replicated copies of the synthetic environment database; each were constrained to content, formats, and structures specific to each client-device. As a result, the potential for correlation errors abounded. The CDB Specification resolves this by defining a single, usable in real-time, open, synthetic environment database representation.

The CDB “runtime publishing” paradigm now permits the simulator vendor the means to not only control client-device load but to globally re-examine and control

the level of correlation within a simulator (or across simulators). While the CDB Specification does not provide explicit jurisdiction over the implementation of this mechanism in the client-devices and/or publishers, it is nonetheless possible to improve parametric correlation, at runtime, via control of the client-devices/publishers.

This topic is discussed in more detail in Section 1.6.6, Synthetic Environment Database Correlation.

### **1.5.5 Increased Synthetic Environment DB Longevity**

The longevity of synthetic environment databases for use in simulation has traditionally been a source of considerable aggravation within the user-community.

At the source-level, the issue was partially resolved through the inception of standards such as SEDRIS SIF. These standards largely isolated the synthetic environment source data from the tools and platforms. As a result, source-level synthetic environment databases are no longer subject to obsolescence when vendors abandon a tool/platform. The advent of such Specifications permitted the modeler to abstract the source-level synthetic environment database from the tools and platforms that are used to create the synthetic environment database.

While this is a step in the right direction, it must be realized that a considerable level of effort (both human and machine) is required to adapt source-level data to a form that is directly usable by each of the simulator client-devices, also known as the runtime-level vendor-specific database format; this is referred to as the “compilation” process. More often than not, the source data is manipulated and adapted to constraints imposed by one or more simulator client-devices. In most cases, the content, resolution and density of synthetic environment databases are repeatedly adjusted to closely match the capabilities and performance of the targeted client-devices. While it is true that the native tool format database remains independent of the targeted client-devices, it is clear that content of the source-level database roughly corresponds to the capabilities of the then-current client-devices. As a result, source-level databases become quickly outdated and do require a complete rebuild to take advantage of new simulator capabilities.

The CDB Specification avoids these pitfalls because the CDB need not be adapted to the constraints imposed by simulator client-devices; that role is relegated to the runtime publishers. Hence, the synthetic environment database can be built, right from the start, to a level of fidelity commensurate with the anticipated useful life of the targeted simulator(s).

### **1.5.6 Reduced Synthetic Environment DB Storage Infrastructure Cost**

For equivalent synthetic environment DB content, the CDB Specification offers a significant storage space savings and significant reductions in the required interconnect infrastructure to supply the synthetic environment DB to the simulator(s).

The reduction in equipment and labor can be attributed to the following CDB features:

1. Simulator-wide unique data representation: eliminate duplication of datasets across client-devices.
2. Compression of storage-intensive datasets: provides effective compression of key datasets.
3. Fine-grain versioning:  
CDB is internally versioned. It is possible to revert to prior representations of the SE without restoring stored back-ups of the CDB. Because the underlying mechanism is fine-grained, only in affected geographic areas or datasets of the CDB need to be versioned.

## **1.6 CDB Primer**

The following paragraphs provide the description and focus of the CDB Specification.

### **1.6.1 CDB Specification Data Representation and Organization**

The CDB Specification provides the means of describing all of the feature sets relevant to simulation (such as terrain, 3D objects). These feature sets are logical regroupings of datasets that are used directly by the simulator client-devices.

In its initial implementation, the Specification supports the following representations:

1. Terrain:  
Is a representation of the terrain shape/elevation, raster imagery, surface attribution and other surface characteristics relevant to distributed simulations.
2. Point feature:  
Is a representation of a single location in space or on the earth's surface. It consists of a single <latitude, longitude> coordinate with or without an elevation. When a point feature does not have an elevation, it is deemed to be on the surface of the earth. It is often associated with a 3D model. The information includes point-feature type identification, location, orientation, connectivity, attribution and other characteristics relevant to simulation.
3. Linear feature:  
Is the representation of predominantly man-made multi-segmented line-oriented features conformed relative to the terrain (such as runways, roads, transmission lines, fences). The information includes linear feature type identification, location, orientation, lineal geometry, connectivity, attribution and other surface characteristics relevant to simulation.
4. Areal feature:  
Is a representation of closed area-oriented features conformed relative to the terrain such as forested areas, fields. The information includes areal feature

type identification, location, orientation, 2D geometry, connectivity, attribution and other surface characteristics relevant to simulation.

5. 3D cultural model:

Is a model that is statically positioned on the terrain or bathymetry skin. Cultural models are often a 3D representation of a man-made or a natural object positioned and conformed relative to the terrain. The information includes its geometry, articulations, raster imagery (texture, normal map, light map, etc.), lighting systems, and other characteristics relevant to simulation.

6. Moving model:

Is a model that is not fixed at one location in the synthetic environment database. The simulation host can update the position and orientation of a moving model at every simulation iteration cycle. A moving model is a 3D representation of man-made objects free to move within the CDB. The information includes feature type identification, (vehicle class, type, model, etc.), geometry, articulations, raster imagery (texture, normal map, light map, etc.), lighting systems, connectivity to special effects, attribution and other characteristics relevant to simulation

7. Materials:

Is a symbolic representation of the surface materials for all of the elements contained within the CDB. Client-devices are required to simulate the synthetic environment over different portions of the electromagnetic spectrum: IR (e.g., FLIR, NVG), microwaves (e.g., Radar), audio (e.g., Sonar), etc. The fidelity of the sensor synthetic environment simulation model that runs in each of the client-devices is highly dependent on the richness and completeness of properties that characterize the synthetic environment database in the electromagnetic spectrum of interest.

**NOTE:** One of the primary objectives of this Specification is to provide and integrate all of the data required by all sensor devices, not just IGs producing the Out-The-Window (OTW) scenes. In addition, the mandate of this Specification is to accomplish this objective in a device-independent fashion. The CDB Specification provides a means for client-devices to retain their internal Sensor Synthetic Environment Model (SEM), and yet do so without introducing device dependencies within the CDB synthetic environment. To accomplish this objective, client-device vendors must provide the appropriate SEM properties for the prescribed CDB Base Materials (see Appendix L), since none of the (vendor/device-specific) material properties are stored within the CDB.

8. Navigational data:

Is a representation of ARINC-424 and DAFIF data in the form of NAVAIDs, Airport/Heliport, Runway/Helipad, Waypoints, Routes, Holding Patterns, Airways, Airspace, etc.



In order to represent the above feature sets, the CDB Specification logically organizes its data in mutually exclusive datasets. Furthermore, the CDB Specification is platform and client-device independent; as a result, the internal data representation is free of client-device specifics and more closely aligned to DB structures/formats supported by prominent industry standard tools.

The CDB Specification native coordinate system is geodetic (latitude, longitude, and elevation) based on the WGS-84 earth model. All the simulator client-devices can access CDB geospecific information using this convention.

### **1.6.2 CDB Specification Logical Structure**

The CDB Specification is a simulator format for simulation and as such, its storage structure is optimized for simulator client-devices runtime performance. Therefore, the internal storage structure is designed with these specific considerations in mind, i.e., as follows:

1. It promotes efficient real-time CDB data access by the simulator client-devices without degrading their performance. The structure allows simultaneous accesses by all of the various simulator client-devices.
2. It promotes efficient database updates and deployment in order to reduce the deployment of a CDB onto one or more simulators.

To address these objectives, the storage structure geographically divides the world into geodetic tiles (bound by latitudes and longitudes), each containing a specific set of features (such as terrain altimetry, vectors), models (such OpenFlight models, RCS models), which are in turn represented by the datasets (see Figure 1-10: CDB Specification Tile/Layer Structure). The datasets define the basic storage unit used in a CDB. 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. At the database generation facility, it is possible to effect small updates, either at the tile level; the feature set level or the dataset level. Similarly, the storage structure fully supports real-time retrieval of CDB content at the tile level and the dataset level. Finally, the CDB's incremental versioning mechanism allows users to efficiently deploy updates to a CDB; only the files whose content differs from a prior version need be generated and transferred from the database generation facility to the simulation facility.

Another benefit of the CDB Specification storage structure is its ability to support the concurrent generation and deployment.



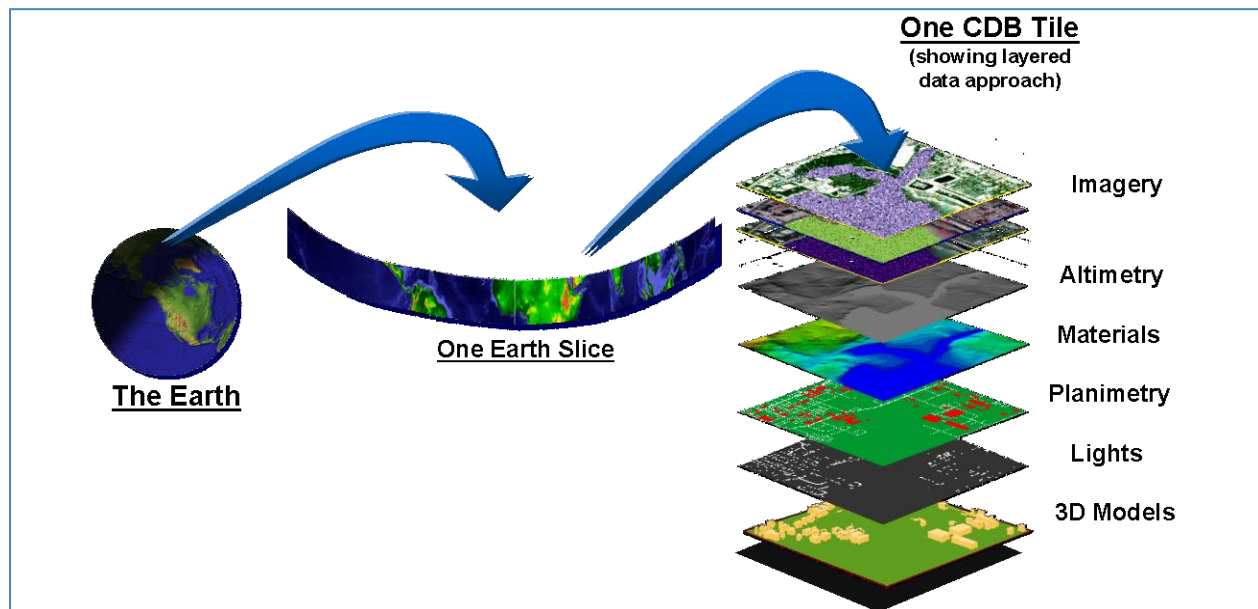


Figure 1-10: CDB Specification Tile/Layer Structure

### 1.6.3 CDB Structure, Organization on Media and Conventions

This CDB Specification describes the CDB data content, representation, as well as the logical organization of the CDB data stored on the CDB server disks:

1. Database Structure:  
Database structure comprises all of the data structures used to access database content (such as file paths, Level-of-Detail (LOD), lists) to speed up access, information layering, versioning and configuration management.
2. Naming Conventions:  
Use to describe a naming hierarchy for cultural models, moving models, lights and model components.

### 1.6.4 Typical Implementation on a Simulator

This section is included here to illustrate a typical implementation of CDB Specification on a flight simulator, when used as an on-line (or runtime) repository. The Specification does not mandate particular simulator architecture or the use of specific computer platforms. The selected implementation varies with the required level of fidelity and performance of the simulator and its client-devices.

**NOTE:** Legacy simulator client-devices can be readily retrofitted for compatibility with the CDB Specification by inserting a runtime publisher in their SE paging pipeline.

As shown in Figure 1-11: Typical CDB Implementation on a Suite of Simulators below, a typical implementation of a CDB Specification System consists of the following main components:

1. **Database Generation Facility (DBGF) and CDB Master Store:**  
A geo-graphically co-located group of workstation(s), computer platforms, input devices (digitizing tablets, etc.), output devices (stereo viewers, etc.), modeling software, visualization software, CDB server, CDB off-line publishing software and any other associated software and hardware used for the development/modification of the CDB. The CDB Master Store consists of a mass storage system (typically a storage array) and its associated network. It is connected to a dedicated DBGF Server.
2. **Update Manager (UM):**  
The Update Manager software consists of both client and server software. The Update Manager Server (UMS) software is located at the DBGF. It manages the CDB updates (versions) and runs in the same platform as the DBGF Server. The Update Manager Client (UMC) software is located at the Simulator Facility and runs on the Update Manager Platform shown in Figure 1-11: Typical CDB Implementation on a Suite of Simulators. It communicates with the UMS to transfer the CDB (partial or complete copy) and its updates.
3. **Simulator Facility CDB Repository:**  
The simulator repository consists of a mass storage system (typically a storage array) and its associated network infrastructure. It is connected to the UMC (primarily for update purposes) and the CDB Servers (for simulator client-device runtime access).
4. **CDB servers:**  
Is an optional<sup>15</sup> gateway to CDB mass storage and applicable infrastructure. The CDB servers access, filter and distribute CDB data in response to requests from the simulator runtime publishers.
5. **Runtime publishers:**  
A term used to describe the computer platforms, and the software that translates and optimizes, at runtime, CDB synthetic environment database to a client-device specific legacy runtime format. Data is pulled from the CDB server and in turn published in response to requests from its attached simulator client-device.
6. **Simulator client-devices:**  
Are simulation sub-systems (IGs, radar, weather server, Computer Generated Forces (CGF) terrain server, etc.) that require a complete or partial synthetic representation of the world. CDB runtime clients may require a CDB runtime publisher to convert the CDB into a form they can directly input.

---

<sup>15</sup> Optionally needed for a large-scale CDB repository whose storage system is based on a Storage Area Network (SAN).

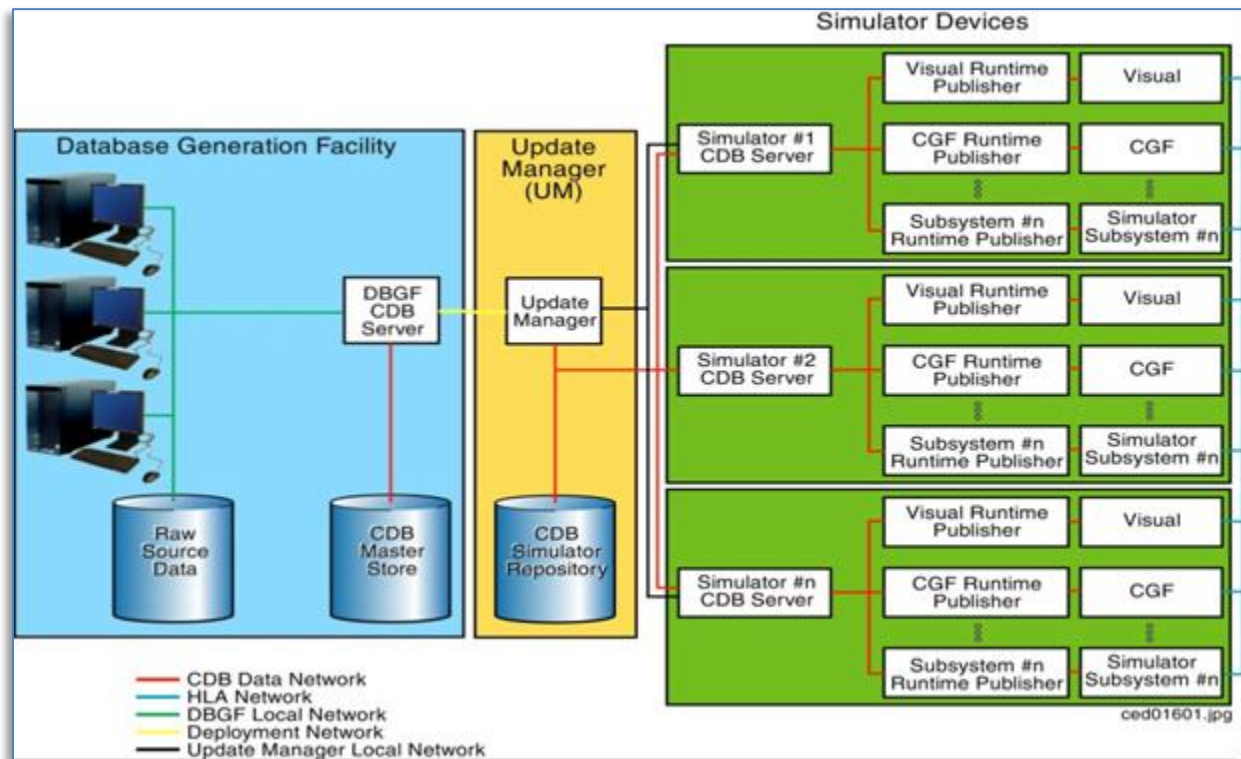


Figure 1-11: Typical CDB Implementation on a Suite of Simulators

#### 1.6.4.1 Database Generation Facility

The DBGF is used for the purpose of CDB creation and CDB updates. Each workstation is equipped with one or more specialized tools. The tool suite provides the means to generate and manipulate the synthetic environment.

#### 1.6.4.2 Database Generation Flow

The CDB considerably simplifies the database generation process, particularly all aspects of database generation that deal with database layering, formatting, structure and level-of-detailing.

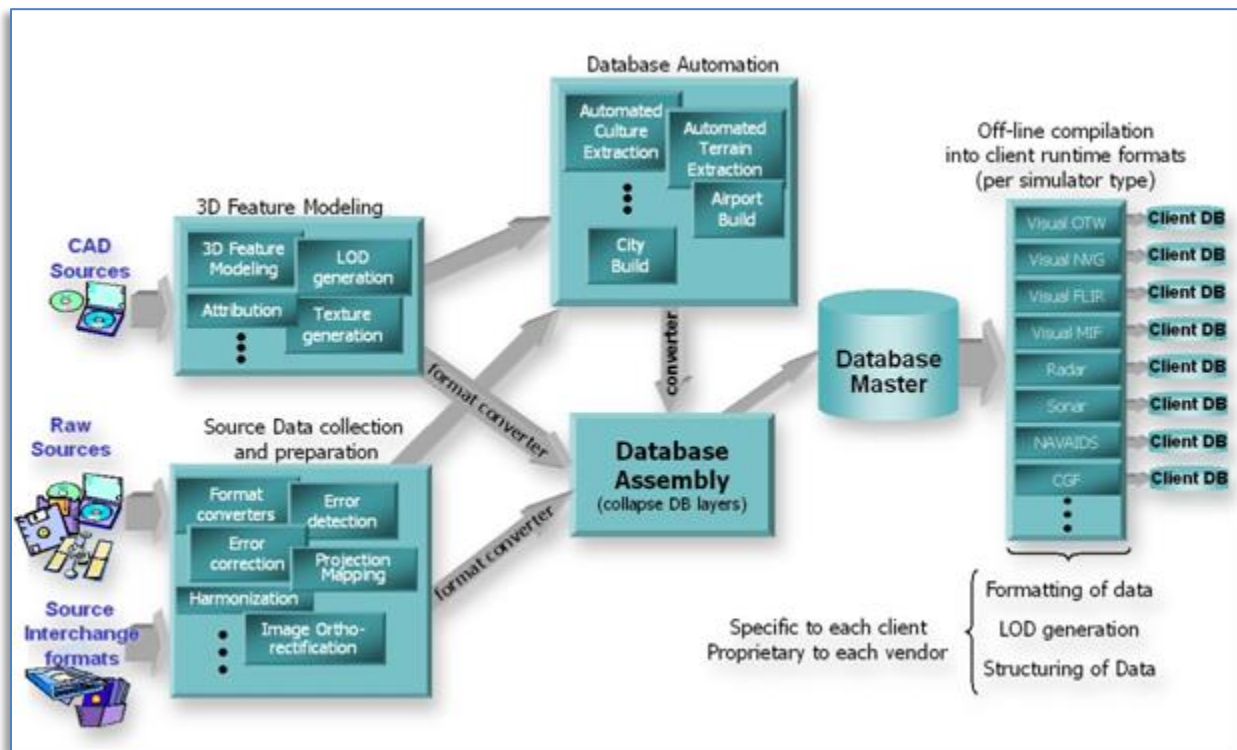


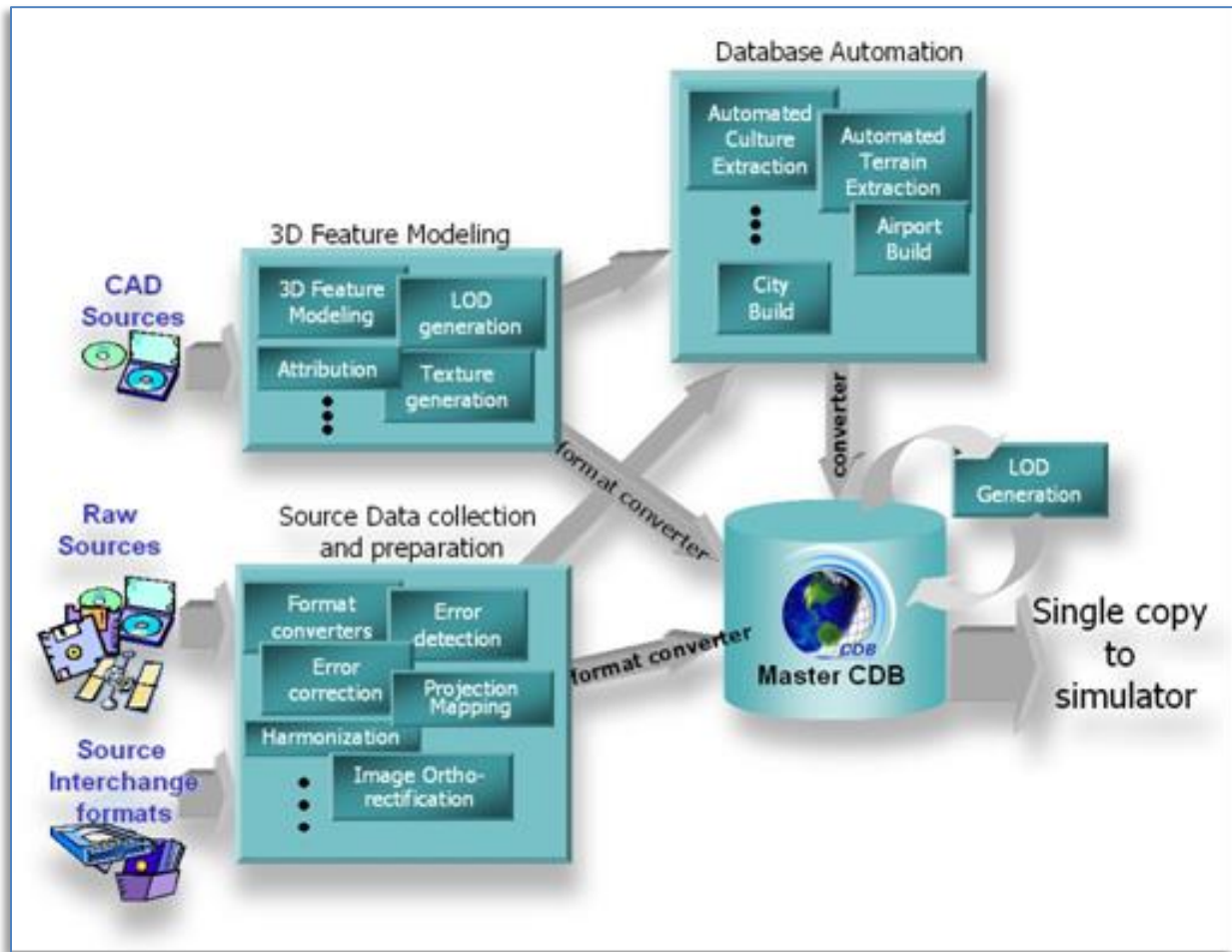
Figure 1-12: Typical DB Generation - CDB Used as DB Repository

Figure 1-12: Typical DB Generation - CDB Used as DB Repository and Figure 1-13: Typical DB Generation Flow - CDB Used as DB & Sim Repository illustrate a typical database generation workflow with the CDB used as a DB workstation repository and the CDB used as a Repository for the DB workstation and the simulator. Both approaches share the same steps, namely:

- **Source data collection and preparation:** This step usually involves the loading of raw (usually) uncorrected data and the conversion to formats native to the database toolset.
- **Source data preparation:** This step usually involves the detection/correction of errors, the harmonization of the data and the correction of errors. In this context, errors signify all instances where the data fails to meet prescribed criteria. For instance, errors can be as straightforward as corrupted digital data. More subtle forms of errors could be textures that fail to meet various brightness, contrast, chrominance, and distortion criteria. Harmonizing data requires that data sources be coherent with each other. An example of non-harmonized dataset is a terrain imagery mosaic built from pictures taken in different seasons, with different illumination conditions, with/without clouds, etc.

- **3D modeling of features:** This step involves the creation of 3D representations for culture features (buildings, trees, vehicles, etc.), the creation and mapping of texture patterns/imagery to the geometrical representation, the generation of the model LOD, and the generation of appropriate attribution data so that the simulator can control the model and have it respond to the simulated environment.
- **Database automation:** Modern database generation tools offer an increasing level-of-automation to the modelers, thereby improving the DB generation timeline (for example, a forest tool that controls the placement of individual trees correlated to the underlying terrain imagery). Over the past few years, tool vendors have introduced a broad set of tools aimed at eliminating highly repetitive modeling tasks; this includes tools for runway generation (including the positioning of stripes, lights, signs, markings, etc.), road/railroad generation, cultural feature extraction from stereo pairs, cultural feature footprint extraction from image classification processes, terrain grid generation from stereo pairs, terrain surface material classification, etc.





**Figure 1-13: Typical DB Generation Flow - CDB Used as DB & Sim Repository**

The result of the above steps yields a group of independent, layered and correlated datasets, (i.e., datasets that are geographically aligned in latitude/longitude (but not always elevation)), all sharing compatible projections, with all of the necessary attribution.

In the non-CDB approach illustrated in Figure 1-12: Typical DB Generation - CDB Used as DB Repository, the correlated layered datasets are collapsed into a single global database during the database compilation process. In many database tools, the generation of the terrain plays a pivotal role in the database assembly process because all of the cultural features are conformed and constrained to the terrain representation and structure. Most client-devices in existence today have interdependent terrain geometry, raster imagery and culture; as a result of this, most tools in use today resolve these inter-dependencies during this critical and computationally expensive database assembly step. The resultant database is stored on disk at the DB generation facility. Following the assembly process, the modeler is required to run an off-line compilation process for each of the supported client-devices. The compilation



process must resolve all of differences between the tool-native representation and the client-device internal representation. During this step, the off-line compiler performs a series of steps that transforms the database into an alternative form (usually proprietary) that resolves:

1. its internal formats
2. its structure and organization (for example, its level-of-detail representations)
3. its naming conventions
4. the number precision and number representation of its data
5. the type and fidelity of data/parameters it uses for its internal algorithms (typically light parameters, FLIR/NVG parameters, etc.)
6. its performance limitations

As stated earlier in Section 1.3.1.1, the CDB can be used as an off-line repository for the authoring tools and as an on-line repository for the simulator(s). In the latter case, there is no need for an off-database compilation process because this step is in-effect performed on-line by the runtime database publishers. Furthermore, since the CDB datasets adhere to data formats commonly in use, most authoring tools in existence today, have the necessary software to read/write data in the mandated formats.

Out of the many steps typically required by the off-line compilation, the CDB only requires that levels-of-detail be generated for the terrain elevation, raster imagery, and the grouping of cultural features. These improvements are expected to yield important savings in man-hrs, machine-hrs and storage when compared to the non-CDB approach.

#### **1.6.4.3 Update Manager**

The creation of the CDB and subsequent updates are performed at the DBGF. The Update Manager (UM) keeps track of these updates and synchronizes the Simulator CDB Repository to the DBGF. The CDB Specification permits flexible and efficient access of the CDB and does so with different levels of granularity; thus, it is possible to perform modifications to the CDB on a complete tile, or on individual datasets of a tile. This permits rapid deployment of the CDB, a feature that is particularly valuable for mission planning and rehearsal. With few exceptions<sup>16</sup>, there is no inter-dependency between datasets and it is possible to modify a dataset (such as the terrain imagery) without reprocessing the complete tile; only the modified dataset requires re-processing. The CDB Specification supports the concurrent creation/modification of the CDB with its deployment. Once a tile, a feature set, or a dataset has been processed, it may be transferred to the simulator facility concurrently with other work performed at the DBGF.

Updates to the simulator CDB repository are performed by the UM. The simulator CDB repository is configured to provide storage for a (partial or complete) copy of

---

<sup>16</sup> The only exceptions to this CDB principle are the MinElevation, MaxElevation datasets which are slaved to the Terrain Elevation dataset and the MaxCulture dataset which is slaved to the GSFeature/GTFeature dataset.

the Database Generation Facility (DBGF) CDB master store. The Update Manager transfers the CDB and its updates by area of interest, allowing for partial updates or even complete copies of the CDB. The Update Manager (UM) simulator CDB repository is used by one or more co-located simulators to retrieve the CDB in real-time.

Additionally, the UM manages the facility's release of the CDB. It maintains versioning information as supplied by the DBGF. Based upon this information, it is possible to request or approve CDB updates to the facility from the UM.

#### **1.6.4.4 CDB Servers**

When the CDB is used as an on-line (or runtime) repository, a set of CDB servers (i.e., the server complex) are required in order to fetch data in real-time from the simulator CDB repository. Each of the CDB servers responds to the requests made by the simulator client-device runtime publishers.

##### **1.6.4.4.1 Runtime Publishers**

When the CDB is used as an on-line (or runtime) repository, a set of runtime publishers are required in order to transform the CDB data into legacy client-devices (simulator subsystems) internal format<sup>17</sup>. The runtime publishers provide a key role in further enhancing overall algorithmic correlation within and across simulators. Each publisher communicates to the CDB server complex and the attached simulator client-device as follows:

1. Receive update requests for synthetic environment data from their respective simulator client-devices.
2. Relays the update request to the CDB server complex.
3. Once the update request is acknowledged and the data retrieved by the CDB server complex, the runtime publisher pulls CDB data from the CDB server complex and converts and formats this data into a form directly usable by the simulator client-device. This processing is accomplished in real-time.
4. Transfers the converted CDB data to the simulator client-device.

##### **1.6.4.4.2 Simulator Client-devices**

The paragraphs below provide a short description of the client-devices found on a typical simulator and the global types of information required from the CDB.

---

<sup>17</sup> Alternately, client-devices can be designed / modified to natively handle the CDB's data model, thereby obviating the need for a separate runtime publishing step.

#### **1.6.4.4.2.1 Visual Subsystems**

Typical visual subsystems compute and display in real-time, 3D true perspective scenes depicting rehearsal and training environments for OTW, IR, simulated Night Vision Goggles (NVG), and 3D stealth IG viewing purposes.

#### **1.6.4.4.2.2 Out-The-Window Image Generator (OTW IG)**

The IG portion of the visual system provides a wide range of features designed to replicate real-world environments. High density and high complexity 3D models can be superimposed onto high-resolution terrain altimetry and raster imagery. Scene complexity with proper object detail and occulting provide critical speed, height and distance cueing. Special effects are implemented throughout the database to enhance the crew's experience and overall scene integrity. Typical IGs optimize the density, distribution and information content of visual features in the scene(s) for all conditions of operations.

The visual subsystem uses time invariant information held in the CDB such as:

1. Terrain altimetry and raster imagery data
2. Cultural feature data
3. Light point data
4. Airport data
5. Material attribution data

#### **1.6.4.4.2.3 Infrared IG**

Included in the CDB coding is the material attribution used by a typical physics-based Infrared Sensor Synthetic environment Model. This model computes, in real-time, the amount of radiated and propagated energy within the simulated thermal bands.

A typical thermal model takes into account the following material properties:

1. Solar absorbance
2. Surface emissivity:  
This coefficient reflects the degree of IR radiation emitted by the surface.
3. Thermal conductivity
4. Thermal inertia:  
This coefficient describes the material ability to gain/lose its heat to a still-air environment.

#### **1.6.4.4.2.4 Night Vision Goggles Image Generation**

Included in the CDB coding is the material attribution (exclusive of any properties) used by NVG simulation models.

#### **1.6.4.4.2.5 Ownship-Centric Mission Functions**

Visual subsystems typically provide a set of ownship-centric Mission Functions (MIF) for use in determining:

1. The Height Above Terrain (HAT), Height Above Culture (HAC), and Height Above Ocean (HAO). This function may report the material type of the texel or the polygon, and the normal of the surface immediately beneath the point.
2. Ownship Collision Detection (CD) with terrain, 3D culture and moving models. This may include long thin objects such as power lines.
3. Line Of Sight (LOS) and Laser Ranging Function (LRF) originating from the ownship. This function may return the range, the material type and the normal of the nearest encountered element in the database. The maximum length of a requested vector is typically limited to the paged-in database.

The mission functions provided by an IG base their computations on a database that has LOD representations equivalent to those used by OTW IGs. Since the visual subsystem scene management mechanisms are essentially slaved to the ownship's position, the terrain accuracy (e.g., its LOD), the cultural density/LOD and the texture resolution decrease with distance from the ownship. As a result, the IG-based mission functions computations are best suited for ownship functions. In cases where the database needs to be interrogated randomly anywhere in the gaming area, simulator client-devices such as Computer Generated Forces (via a terrain server) are best suited because their architecture is not ownship-centric.

#### **1.6.4.4.2.6 Computer Generated Forces (CGF)**

CGF provides a synthetic tactical environment for simulation-based training. It simulates behaviors and offers interactions between different entities within the simulation. It models dynamics, behavior doctrines, weather conditions, communications, intelligence, weapons and sensor interactions, as well as terrain interactions. CGF offers modeling of physics-based models in a real-time natural and electronic warfare environment for air, land and sea simulations.

Typically, CGF is able to create a realistic simulated multi-threat, time-stressed environment comprising items such as:

1. Friendly, enemy and neutral entities operating within the gaming area
2. Interaction with weather conditions currently in the simulation
3. Entities with representative dynamics (velocity, acceleration, etc.), signatures, vulnerabilities, equipment, communications, sensors, and weapons
4. CGF uses time invariant information held in CDB such as:
  - a. Terrain altimetry and raster imagery
  - b. Cultural features
  - c. Linear (vector) and areal information

- d. Sensor signatures
- e. Moving Models

#### **1.6.4.4.2.7 Weather Simulation**

The Weather Simulation (WX) involves computing and analyzing the various weather components and models around important areas defined in a simulation, in order to produce realistic real-life scenarios for the sub-systems being affected by weather effects. As such, a weather data server typically handles the weather simulation; this server handles requests for weather-related data such as temperature, 3D winds, turbulence gradients, and complex weather objects such as clouds, frontal systems or storm fronts.

WX uses time invariant information held in CDB such as terrain elevation and (potentially) significant features with 3D modeled representations to compute weather and wind patterns.

#### **1.6.4.4.2.8 Radar**

Typical Radar Simulation Models require modeling of all real-life and man-made effects or objects that can cause significant echo returns from the wavelengths of the simulated Radar RF main beam and side lobes. Additionally, LOS computations are necessary for proper target occultation by the Radar.

The Radar subsystem uses time invariant information held in CDB such as:

1. Terrain altimetry and Raster materials
2. Cultural features with either 2D and 3D modeled representations
3. Material properties
4. Land/Coastline/Man-Made features
5. Target shapes (RCS polar diagrams, 3D models)

#### **1.6.4.4.2.9 Navigation System**

The Navigation System provides the navigation information around the areas and routes as defined in a simulation in order to provide precise NAVAIDs data which will generate well correlated sub-systems being part of such simulation scenarios.

As such, the Navigation System Simulation handles navigation aids information requests from other simulator client-devices such as:

1. Tactical Air Navigation (TACAN)
2. Automatic Direction Finder (ADF)
3. VHF Omni Range (VOR)
4. Instrument Landing System (ILS)
5. Microwave Landing System (MLS)
6. Doppler Navigation System (DNS)
7. Global Positioning System (GPS)

8. Inertial Navigation Unit (INU)
9. Non-Directional Beacons (NDB)

In addition to the NAVAIDs, the navigational data include datasets such as:

1. Communications Stations data
2. Airport/Heliport (including SIDs, STARs, Terminal Procedure/Approaches, Gates)
3. Runway/Helipad
4. Waypoints
5. Routes
6. Holding Patterns
7. Airways
8. Airspaces

NAV uses time invariant information held in CDB such as:

1. ICAO code and Airport Identifier
2. NAVAIDs frequency, channel, navigational range, power
3. Declination
4. Magnetic variations
5. Communications Stations data
6. Airport/Heliport
7. Runway/Helipad

#### **1.6.4.5 Other Applications of the CDB Specification**

While the Specification was initially targeted primarily for use in flight simulator applications, it is entirely suited to a broad range of other applications that make use of the same datasets; these include (and are not limited to):

1. Ground warfare simulation
2. Anti-submarine warfare
3. Visualization
4. Modeling and Simulation
5. Urban Planning
6. Natural Resource Management
7. Emergency Management

The implementation of the CDB Specification on legacy simulator client-devices usually mandates the use of the runtime publishers; these costs are largely offset by the consolidation (and substantial reduction) of storage and associated infrastructure. In the future, it is anticipated that the runtime publishing computer infrastructure will be largely “absorbed” by higher performance client-devices that will natively process the CDB without an intermediate conversion into a legacy internal format.

In applications that are less demanding than flight simulation, the CDB Specification can be implemented without resorting to high-performance computer platforms. For instance, the simulator repository mass storage system (shown in Figure 1-11: Typical



CDB Implementation on a Suite of Simulators) can be a single disk storage device. In demanding applications, the simulator repository can be implemented as a large-scale Storage Area Network (SAN). Similarly, the UM, the CDB server and each of the client-device runtime publishers can be implemented as software tasks within a single computer platform, or can even be migrated to software tasks running internally within the client-devices. Figure 1-8: Typical Implementation of CDB Specification on Desktop Computer, illustrates a desktop trainer consisting of a Stealth Viewer and radar simulations, implemented as software applications running on a single computer platform; both applications pull their synthetic environment data from a disk-resident CDB.

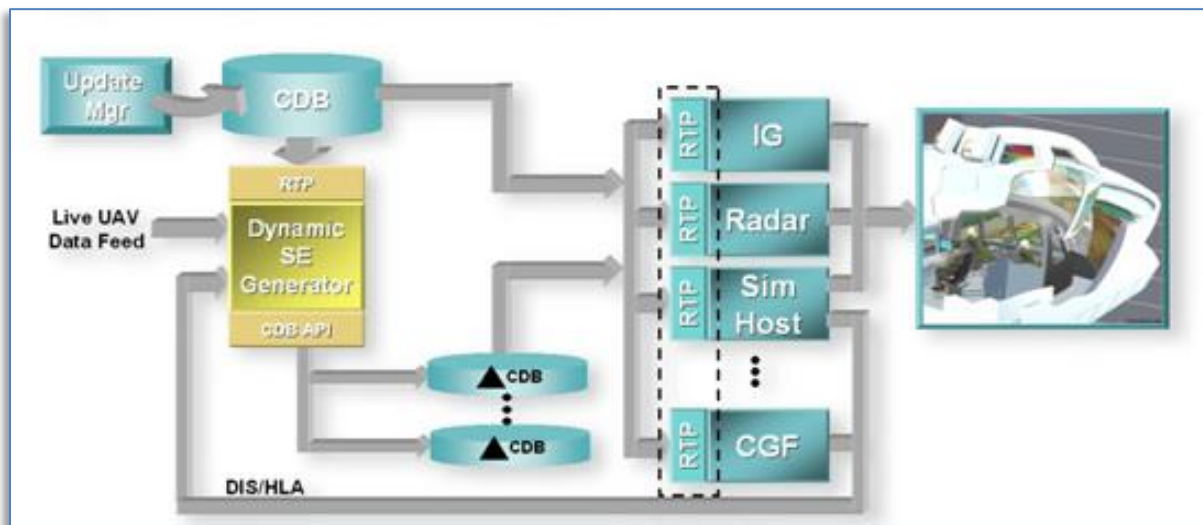
### 1.6.5 Use of CDB in Applications Requiring Dynamic Synthetic Environments

A CDB-compliant simulator already incorporates most of the architectural features required to support the handling of Dynamic Synthetic Environments (aka DSE). Figure 1-14: Versioning Paradigm Applied to Dynamic SE, illustrates how CDB-compliant simulator architecture can be extended to permit the runtime modification of the CDB. This can be accomplished by leveraging simulator architectures that natively adhere to the CDB data schema and to the CDB versioning capabilities.

This capability requires that the simulator be equipped with a Dynamic Synthetic Environment Generator whose interfaces conform to the CDB Specification; all other elements of the simulator architecture are identical to those of a CDB-compliant simulator.

Two applications of this principle could be envisaged:

1. ***DB update from DIS:*** This is the so-called “Dynamic Terrain/Synthetic Environment”, where a group of confederates interoperate over a DIS network by receiving update commands that trigger runtime modification of the CDB.
2. ***DB creation/update from live data feed:*** Terrain areas could be created or modified on-the-fly, based on live data (such as terrain imagery or LIDAR data transmitted by a UAV) and provided instantly to the confederates.



### 1.6.6 Synthetic Environment Database Correlation

Synthetic environment correlation issues arise whenever two or more devices exhibit a discernible level of informational inconsistency. Correlation of synthetic environment databases has long been a problem in simulation, primarily aggravated by the fact that each of the simulator client-devices used distinct runtime databases.

There are many different types of correlation errors, and each have a contribution to the overall correlation problems that has plagued simulators for several years. They are defined here as follows:

1. **Raw source correlation:** Is the degree of informational consistency between two or more sets of raw data<sup>18</sup> (i.e., inputs to a modeling station) representing aspects of the same environment (for instance, the correlation errors arising from Digital Terrain Elevation Data (DTED) elevation data that does not perfectly match to satellite raster imagery due to oblique view distortions induced by the satellite). Correlation errors are intrinsic to the process of gathering data because since there is no means to gather all of the required data from a single device, at a single instant in time. Instead, datasets (e.g., elevation, raster imagery, geometry) are each gathered from various devices of various types (with distinct precision, formats, capabilities, fidelity) at different times. This in turn leads to a broad range of correlation errors typically resolved by the modeler during the final assembly of the synthetic environment from its sources.

<sup>18</sup> In this context, raw source denotes any input to the modeling workstation that is used to assemble the synthetic environment; consequently, the data may have undergone some level of post-processing (such as image color-balancing, image ortho-rectification, etc.) or may be in a specialized source interchange format (such as SIF, SEDRIS, etc.).

2. **Source database self-correlation:** Is the degree of informational consistency between the internal datasets of a source database produced by a DB generation toolset. To a large extent, the effort expended at DB generation time consists in eliminating (or at least reducing) correlation errors arising from miss-correlated raw source data.
3. **Runtime database correlation:** Is the degree of informational consistency between two or more runtime client-specific databases representing the same synthetic environment<sup>19</sup>. The likelihood of achieving correlated runtime client-device databases is particularly low when different authoring tools (and possibly different source data) are used to assemble each of the compiled runtime databases. In recent years, some authoring tools have been improved to automatically produce a set of client-device database from one common repository (internal to the tools). Nonetheless, it is still current practice within the simulation community to independently deploy the simulator client-device databases; as a result, correlation errors may occur especially if the master database repository is constantly evolving. The CDB Specification eliminates database correlation errors since only one database is used to represent the same synthetic environment. The CDB is a single database that can be accessed simultaneously by all simulator client-devices at runtime. By definition, it addresses all runtime database-level correlation errors.
4. **Numerical accuracy correlation:** Is the degree of informational consistency between the outputs of two or more devices, with each device performing the same algorithms, using the same control parameters but performing internal computations to a different numerical accuracy. Consider for example two devices computing the sin of an angle, one with a series of 10 terms, and another with an interpolation of a look-up table with 100 entries, or one device using 32-bit signed integers for its internal computations and the other using single-precision floats. The CDB Specification reduces numerical accuracy correlation errors because a single representation is used for each dataset. Note however that numerical correlation may deteriorate due to the variances in numerical precision inherent to each client-device.
5. **Algorithmic correlation:** Is the degree of informational consistency between the outputs of two or more devices, with each device performing its internal computations to the same numerical accuracy, but using different algorithms with possibly different control parameters. Consider for example, two devices meshing terrain from a regular grid of elevation points, one using a regular mesh of right-handed triangles using the elevation points as vertices, and the other with a DeLauney triangulated mesh derived from the grid of elevation points. Algorithmic correlation errors can be introduced anywhere in the dataset processing chain, ranging from the raw source level right through to the internals of the simulator client-devices. While it is not possible to

---

<sup>19</sup> A runtime client-specific database is a device-loadable database format that can be processed by a target device.

mandate complete algorithmic uniformity throughout this elaborate chain, the CDB Specification offers solutions to this issue<sup>20</sup>. Firstly, only one database is produced, so all DB authoring tool algorithmic correlation issues are eliminated. Note that the implementation of runtime publishers on a simulator can play a role in improving overall algorithmic correlation<sup>21</sup>.

6. ***Parametric correlation:*** Is the degree of informational consistency between the outputs of two or more devices, with each device performing the same algorithms with identical numerical precision with different control parameters (e.g., consider two devices generating regular meshes of right-handed triangles based on a regular grid of elevation points organized by LOD, one using an LOD meshing tolerance parameter of 1m and the other one using 2m). Note that it is clearly possible to create a synthetic environment database whose content is at a level of resolution, fidelity and accuracy that can overwhelm any client-device; however, the amount and type of data that is rejected by each type of client-device can vary considerably. For example, a NAVAIDS data server client-device need only be concerned with Digital Aeronautical Flight Information File (DAFIF™), and as a result is virtually unaffected by changes in the resolution of the terrain elevation data which could easily increase terrain SE content by 100-fold. The implementation of the CDB Specification on a simulator can reduce and/or eliminate parametric error by exercising explicit control, at run-time, on the resolution of the data processed by the client-devices. This approach is much preferable to off-line published approaches where parametric correlation is established during the compilation of the runtime databases and cannot be changed once the runtime database is produced.

Table 1-1: Summary of Synthetic Environment Database Correlation Errors, provides a summary of the different types of correlation errors and how/where such errors can be addressed. Figure 1-15: Sources of Synthetic environment Database Correlation Errors, illustrates the conventional synthetic environment database process from raw source, the assembly of datasets at the DB workstation, the publishing into runtime databases, and the rendering by the simulator client-devices.

---

<sup>20</sup> For instance, the CDB does not impose the terrain skinning and meshing conventions that each of the runtime publisher generates for its attached client-device. However, it is understood that if all client-devices would use the CDB meshing conventions natively, correlation issues between clients would be significantly reduced.

<sup>21</sup> Since each dataset is uniquely represented, it is possible to share a greater number of algorithms between CDB runtime publishers. This ensures that datasets are processed in an identical fashion whenever two or more publishers share the same algorithm.

**Table 1-1: Summary of Synthetic Environment Database Correlation Errors**

Correlation		Description	Addressed by	CDB Spec	Sim Devices	Database Tools
Data	Raw Source	Raw source correlation errors caused by data collections at different times and from different devices. Also caused by registration errors.	Toolset and human operators address raw source correlation errors ensuring that data is properly corrected and registered.			X
	Datasets	Dataset correlation errors are caused by discrepancies between co-located dataset layers.	Toolset ensures that dependent dataset layers are updated together, while CDB Specification minimizes these dependencies.	X		X
	Runtime Sources	Information from several runtime databases contains conflicting information.	CDB Specification addresses runtime source correlation by enforcing a single runtime database.	X		
Computational	Algorithmic	Different client-devices may use different algorithms to filter information and to simulate real-world device.	Simulation clients address algorithmic correlation errors. They ensure that the different algorithms used are compatible.		X	
	Parametric	Same client-devices may run with different initial parameters.	A simulation that would ensure each client-device runs at just discernible error levels.		X	
	Accuracy	Different computational platforms may run at different numerical accuracies.	Using the same platform with identical numerical accuracy for all client-devices and servers. Developing software according to strict rules and guidelines addresses computational accuracy correlation errors.	X	X	X



<b>Correlation</b>		<b>Description</b>	<b>Addressed by</b>	<b>CDB Spec</b>	<b>Sim Devices</b>	<b>Database Tools</b>
Temporal	Synchronism	Lags and delays introduced by networking systems as well as subsystems using different time bases may cause subsystems to be unsynchronized.	Through system architecture, system is designed to use proper bandwidths and computational methods to reduce latencies. All simulator client-devices use the same clock.		X	
	Paging latency	Paging and runtime publishing may introduce delays long enough to prevent some simulation clients to get information on time.	Paging and runtime publishing tasks and client-device paging software ensures that runtime information is available on time.		X	



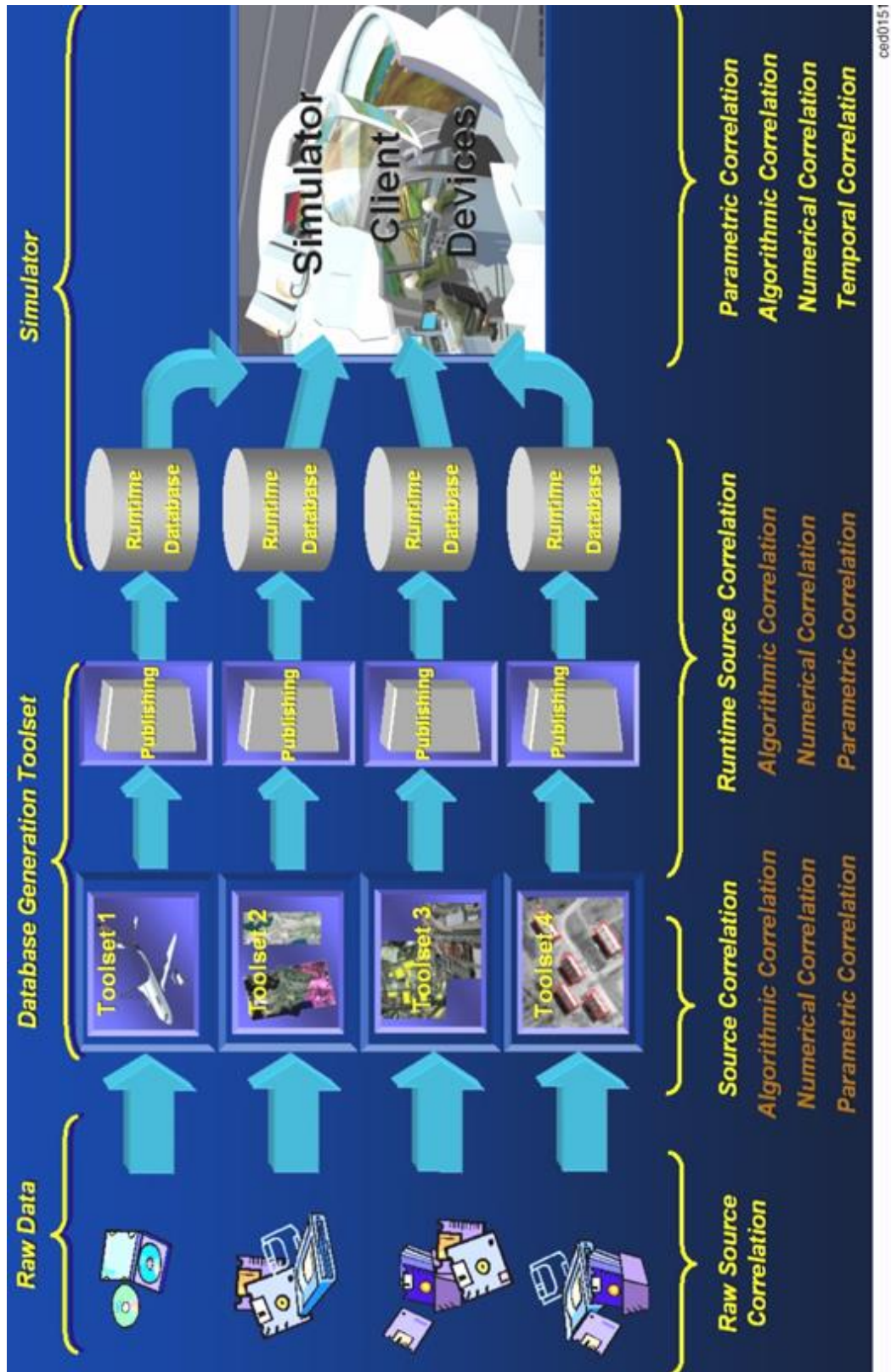


Figure 1-15: Sources of Synthetic environment Database Correlation Errors

The implementation of the CDB Specification on a simulator can also provide the means to reduce and control the sources of parametric correlation at the client-device level. The underlying cause of parametric correlation usually points to the intrinsic capabilities (i.e., the performance and functionality) of each client-device. Since many client-devices are unable to cope with the synthetic environment database at its full fidelity, the off-line compilers “filter-out” portions of the database (e.g., level of resolution, level of fidelity) in order to meet the limited functionality or the real-time constraints of the targeted device. In order to further control processing variations, some client-devices are equipped with the means to dynamically “filter-out” portions of the database in order to meet real-time constraints. The filters can be typically controlled statically (and sometimes even dynamically) through the use of parameters.

The amount and type of data that is rejected by each type of client-device can vary considerably. This is the underlying cause of parametric correlation errors. In conventional simulator client-devices, the handling of parametric correlation is handled in a half-hazard fashion, largely because the “filter parameters” are either fixed or inaccessible to the user. Distinct database compilers generate distinct runtime databases, each configured with their respective filtering parameters.

The CDB Specification offers multi-tiered solutions to this problem. Firstly, since it has a unique data representation, the resolution, fidelity and accuracy of the synthetic environment database, as seen by each of runtime publisher attached to the client-devices, is completely correlated. Secondly, since the publishing process is done on-line, it is possible to provide the user access to the filter parameters so that he can globally control resolution, fidelity and accuracy (and hence correlation) of the synthetic environment database across all simulator client-devices. While the CDB Specification does not provide explicit jurisdiction over the implementation of this mechanism in the client-devices/publishers, it is nonetheless possible to improve parametric correlation, at runtime, via control of the client-devices/publishers. This new paradigm now permits the simulator user the means to not only control client-device load but to globally re-examine and control the level of correlation within a simulator (or across simulators).

The control of parametric correlation requires a working understanding of the characteristics of the client-device. At a minimum, one must consider the operating limits of all client-devices and ensure that the runtime publishers limit synthetic environment content to the limits imposed by the client-devices<sup>22</sup>. Secondly, one must have a working model of the “filter parameters” made available by the client-devices or by their runtime publishers. Thirdly, one must have a working performance model of each client-device. Finally, one must have an understanding of “just discernible threshold” of correlation as it applies to each client-device. For example, it is unlikely that increasing the terrain resolution from 1m to ½m would produce a discernible change in how a CGF system allows simulated players to

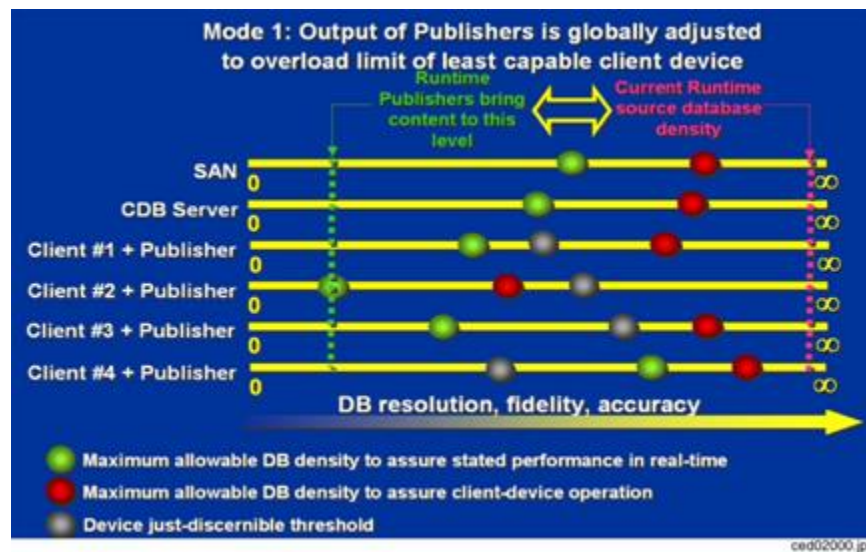
---

<sup>22</sup> For instance, an IG might refuse to operate if confronted with an area modeled with 1 millimeter OTW texture.

interact realistically with one another. As a result, the ½m terrain resolution of the CGF system is below the “just discernible threshold” of that device. The ½m data can be discarded without reducing the level of correlation provided by that client-device.

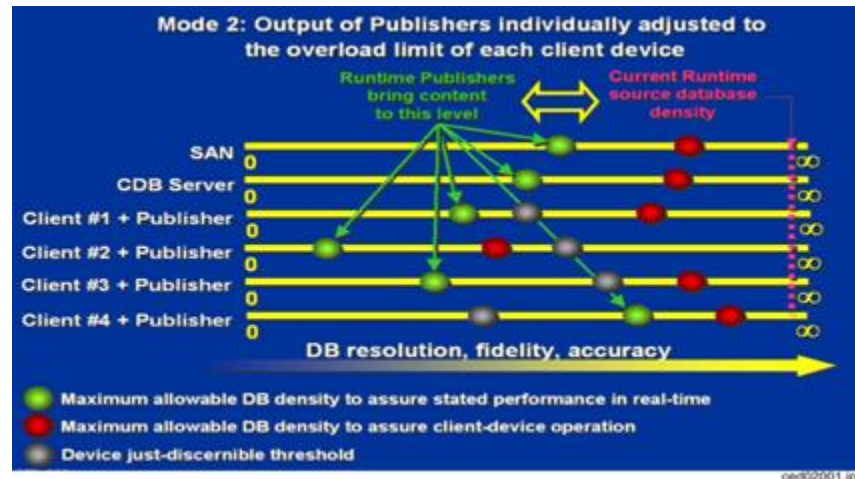
As mentioned earlier, the runtime publishing paradigm permits the simulator user the means to control client-device load and to globally re-examine and control the level of correlation within a simulator (or across simulators). Several choices are possible, depending on the flexibility offered in the implementation of the runtime publishers and the client-devices. The choices are:

1. Publishers globally adjusted to overload limit of least capable client-device (see Figure 1-16: Overload Limit of Least Capable Client-Device):



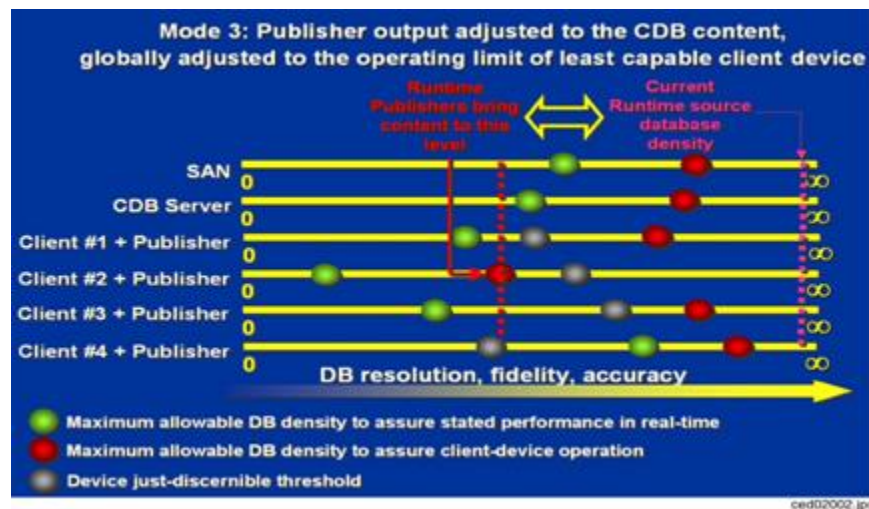
**Figure 1-16: Overload Limit of Least Capable Client-Device**

- a. Assuming sufficient (parameter) controls in each of the client-devices, parametric correlation errors can be eliminated by setting the filter parameters of all client-devices/publishers to the least-capable client-device in the simulator.
  - b. Success is contingent on suitable set of parameters in each client-device.
  - c. The result is full correlation with no overloads, but with the lowest observed DB fidelity.
2. Publishers individually adjusted to the overload limit of each client device (see Figure 1-17: Overload Limit of Each Client-Device):



**Figure 1-17: Overload Limit of Each Client-Device**

- a. This is the approach used in virtually all simulators in operation today.
  - b. Result is partial correlation with no overloads, and high-observed DB fidelity.
3. Publishers adjusted to the CDB content, but globally adjusted to the operating limit of the least capable client-device (see Figure 1-18: Operating Limit of Least Capable Client-Device):

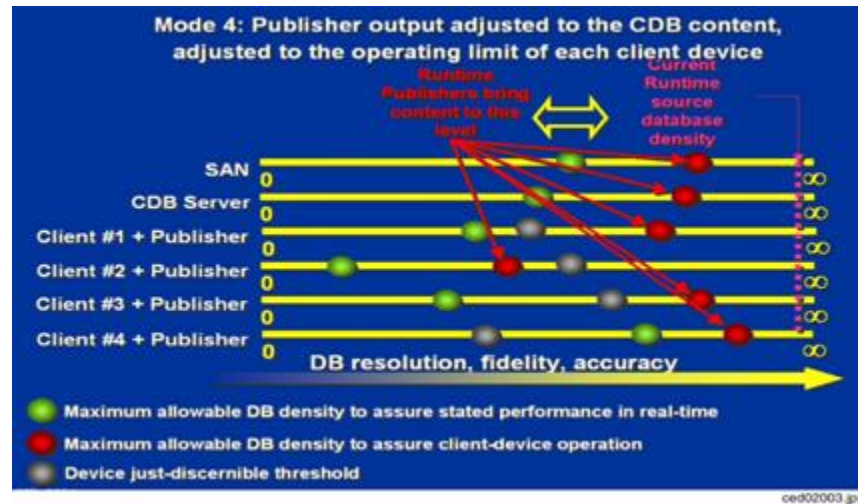


**Figure 1-18: Operating Limit of Least Capable Client-Device**

- a. Client-devices are allowed to overload in areas where database content is deemed critical.
- b. Result is full correlation, possible overloads with high-observed DB fidelity.

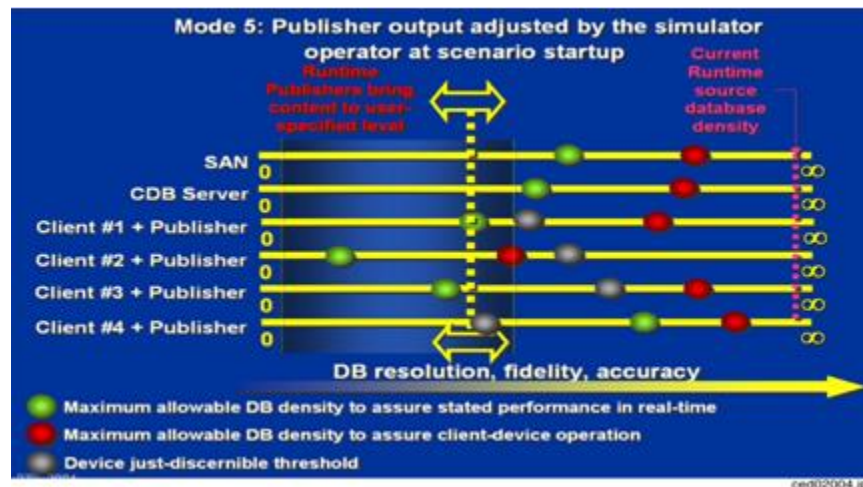


4. Publishers adjusted to the CDB content, but individually adjusted to the operating limit of each client-device (see Figure 1-19: Individually Adjusted to the Operating Limit of Each Client-Device):



**Figure 1-19: Individually Adjusted to the Operating Limit of Each Client-Device**

- a. Client-devices are allowed to overload in areas where database content is deemed critical.
  - b. Result is, possible overloads and the highest observed DB fidelity.
5. Publishers adjusted by the simulator operator at scenario startup (see Figure 1-20: Adjusted by the Simulator Operator at Scenario Startup):



**Figure 1-20: Adjusted by the Simulator Operator at Scenario Startup**





## Chapter 2

### 2 CDB Concepts

This chapter presents basic CDB concepts of the Specification. These concepts are either reused by other concepts or used repeatedly throughout the Specification.

#### 2.1 Partitioning the Earth into Tiles

The CDB tiling approach has the following characteristics:

1. The earth model is divided (in latitude) into slices.
2. The slice's x-axis is aligned to WGS-84 lines of latitude.
3. The slice's y-axis is aligned to WGS-84 lines of longitude.
4. The number of units along the slice's y-axis for a given level of detail is the same for all slices. The earth surface geodetic dimension in arc-second of y-axis units within an earth slice is exactly the same, regardless of latitude.
5. The geodetic dimension of an x-axis unit in arc-second is constant within a zone, but is re-defined at pre-selected latitudes to achieve a greater level of spatial sampling uniformity in all tiles; this overcomes the narrowing effect of increased latitudes on longitudinal distances. The definition of zones in the CDB is the same as those in DTED (with the exception of the poles).
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  $2^n$  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. At this point we introduce the concept of a CDB Geocell, which differs slightly from a DTED cell. Both DTED and the CDB have Geocells whose height is 1 degree but whose width varies depending on its latitude. The only difference is that the CDB provides for an additional zone at each of the poles. Table 2-2 shows the dimensions of a CDB Geocell per zones of latitude. For instance, in latitude zone 5, which goes from  $-50^\circ$  to  $50^\circ$  of latitude, a CDB Geocell is  $1^\circ \times 1^\circ$ , in zone 4 and 6 which goes from latitude  $50^\circ$  to  $70^\circ$  the cell size is  $1^\circ \times 2^\circ$ . The main reason to introduce this concept is to maintain a reasonable eccentricity between the sides by trying to keep them as close to a square as possible. Variable CDB Geocell size in the CDB Specification 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 criteria are used to define the size of a CDB Geocell:

- a. A CDB Geocell must contain a whole number of DTED Geocells; in other words a CDB Geocell must start and end on a whole degree along the longitudinal axis. This is done so as to facilitate mapping from CDB Geocells to DTED Geocells,
- b. The length of the CDB Geocell must be a whole factor of 180, in other words 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.

**Table 2-1: Intervals for DTED Level 2**

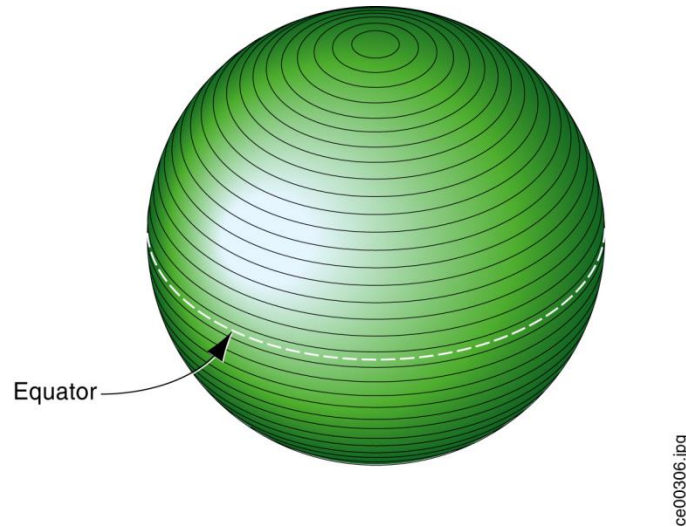
DTED Zone	Latitude Range (Degrees)	Latitude Interval (Arc seconds)	Longitude Interval (Arc seconds)
I	0 – 50 N-S	1	1
II	50 – 70 N-S	1	2
III	70 – 75 N-S	1	3
IV	75 – 80 N-S	1	4
V	80 – 90 N-S	1	6

**Table 2-2: Size of CDB Geocell per Zone**

CDB Zone	Latitude Range (Degrees)	CDB Geocell size (° Lat × ° Lon)	Number of DTED Geocells
0	$-90 \leq \text{lat} < -89$	1 × 12	12
1	$-89 \leq \text{lat} < -80$	1 × 6	6
2	$-80 \leq \text{lat} < -75$	1 × 4	4
3	$-75 \leq \text{lat} < -70$	1 × 3	3
4	$-70 \leq \text{lat} < -50$	1 × 2	2
5	$-50 \leq \text{lat} < +50$	1 × 1	1
6	$+50 \leq \text{lat} < +70$	1 × 2	2
7	$+70 \leq \text{lat} < +75$	1 × 3	3
8	$+75 \leq \text{lat} < +80$	1 × 4	4
9	$+80 \leq \text{lat} < +89$	1 × 6	6
10	$+89 \leq \text{lat} < +90$	1 × 12	12

### 2.1.1 Description

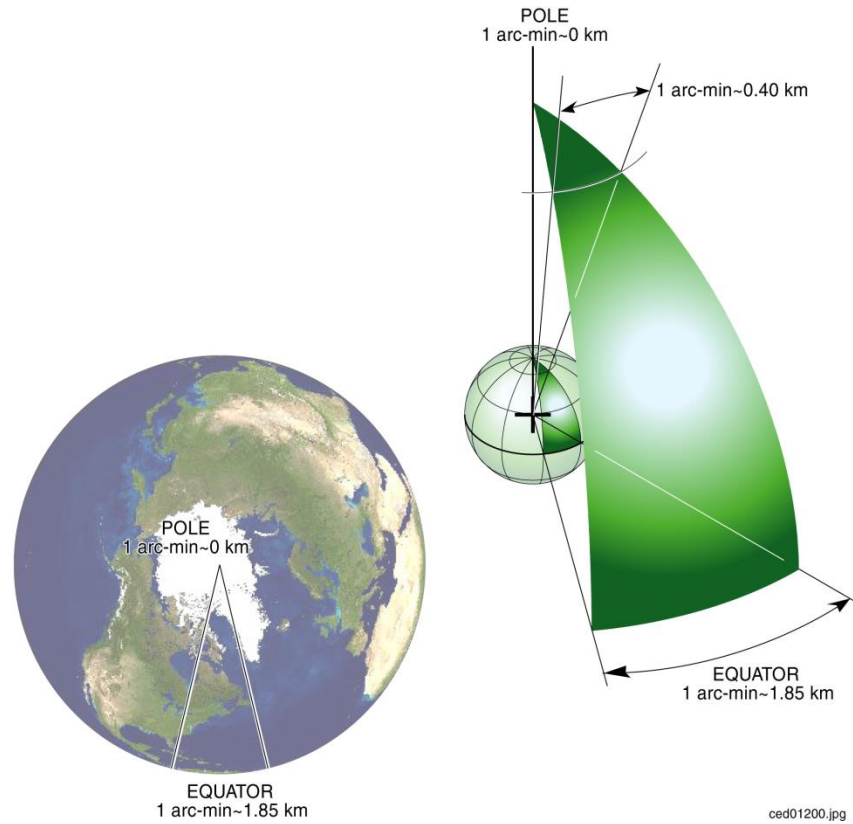
The CDB Specification represents the earth as a fixed number of slices divided equally along a latitude axis as illustrated in Figure 2-1: CDB Earth Slice Representation.



**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. 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-second. The  $x$ -axis and  $y$ -axis fixed increment unit are referred hereafter a  $XUnit$  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 (meter). This property naturally leads to define 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 Specification 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 versus Latitude**

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

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

The CDB Specification 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 slide is one degree.

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

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

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

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

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

*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-LODs)

Since the CDB Specification defines *NbXUnitInCDBGeocell* and *NbYUnitInCDBGeocell* 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).

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

$$YUnit_{LOD} = \frac{YUnit}{2^{LOD}}$$
(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).

$$NbXUnitInCDBGeocell_{LOD} = NbXUnitInCDBGeocell \times 2^{LOD}$$

$$NbYUnitInCDBGeocell_{LOD} = NbYUnitInCDBGeocell \times 2^{LOD}$$

$$TotalNbUnitInCDBGeocell_{LOD} = NbXUnitInCDBGeocell_{LOD} \times NbYUnitInCDBGeocell_{LOD}$$
(2–6)

The CDB standardizes tile sizes to  $1024 \times 1024$  (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 the CDB Specification limits each earth slice 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 the CDB 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 vs Tile and Grid Size**

<b>CDB LOD</b>	<b>Grid Size (n × n)</b>	<b>Approximate Tile Edge Size (meters)</b>	<b>Approximate Grid Spacing (meters)</b>
–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	$1.11319 \times 10^{+05}$	$4.34840 \times 10^{+02}$



CDB LOD	Grid Size (n × n)	Approximate Tile Edge Size (meters)	Approximate Grid Spacing (meters)
-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	$8.69680 \times 10^{+02}$	$8.49297 \times 10^{-01}$
8	1024	$4.34840 \times 10^{+02}$	$4.24648 \times 10^{-01}$
9	1024	$2.17420 \times 10^{+02}$	$2.12324 \times 10^{-01}$
10	1024	$1.08710 \times 10^{+02}$	$1.06162 \times 10^{-01}$
11	1024	$5.43550 \times 10^{+01}$	$5.30810 \times 10^{-02}$
12	1024	$2.71775 \times 10^{+01}$	$2.65405 \times 10^{-02}$
13	1024	$1.35887 \times 10^{+01}$	$1.32703 \times 10^{-02}$
14	1024	$6.79437 \times 10^{+00}$	$6.63513 \times 10^{-03}$
15	1024	$3.39719 \times 10^{+00}$	$3.31756 \times 10^{-03}$
16	1024	$1.69859 \times 10^{+00}$	$1.65878 \times 10^{-03}$
17	1024	$8.49297 \times 10^{-01}$	$8.29391 \times 10^{-04}$
18	1024	$4.24648 \times 10^{-01}$	$4.14696 \times 10^{-04}$
19	1024	$2.12324 \times 10^{-01}$	$2.07348 \times 10^{-04}$
20	1024	$1.06162 \times 10^{-01}$	$1.03674 \times 10^{-04}$
21	1024	$5.30810 \times 10^{-02}$	$5.18369 \times 10^{-05}$
22	1024	$2.65405 \times 10^{-02}$	$2.59185 \times 10^{-05}$
23	1024	$1.32703 \times 10^{-02}$	$1.29592 \times 10^{-05}$

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.

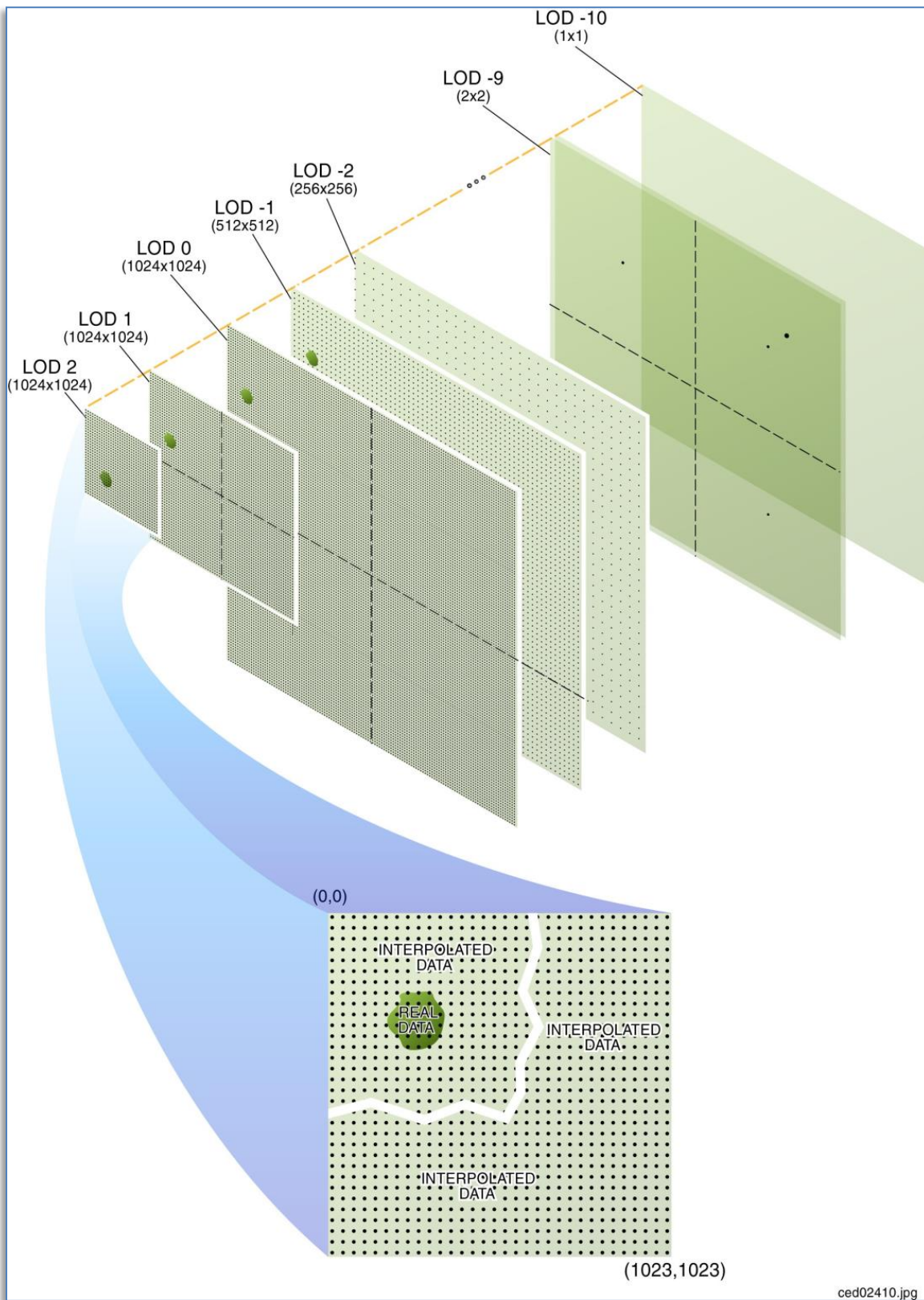


Figure 2-3: Tile-LOD Hierarchy

### 2.1.2.1 Tile-LOD Area Coverage Rules

A tile from LOD -10 to LOD 0 occupies 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 is recursively subdivided into smaller tiles, every level corresponding to a finer representation of the previous level allowing for multiple levels of details. 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, Earth Slice Example (Five Levels-of-Detail), shows an earth slice represented with five distinct LODs. In the illustration, certain CDB Geocell of the earth slice is represented with all five LODs, while others have only three or four LODs. Each Geocell, when present, is represented by an instance of at least the coarsest tile supported by the CDB Specification, i.e., one tile at LOD -10. In addition, if a tile exists at LOD  $n$ , not all 4 tiles at LOD  $n+1$  need to exist. However, if a tile is present at LOD  $n$ , it implies that a coarser tile exists at LOD  $n-1$  covering the area of the tile at LOD  $n$ , until LOD -10 is reached.

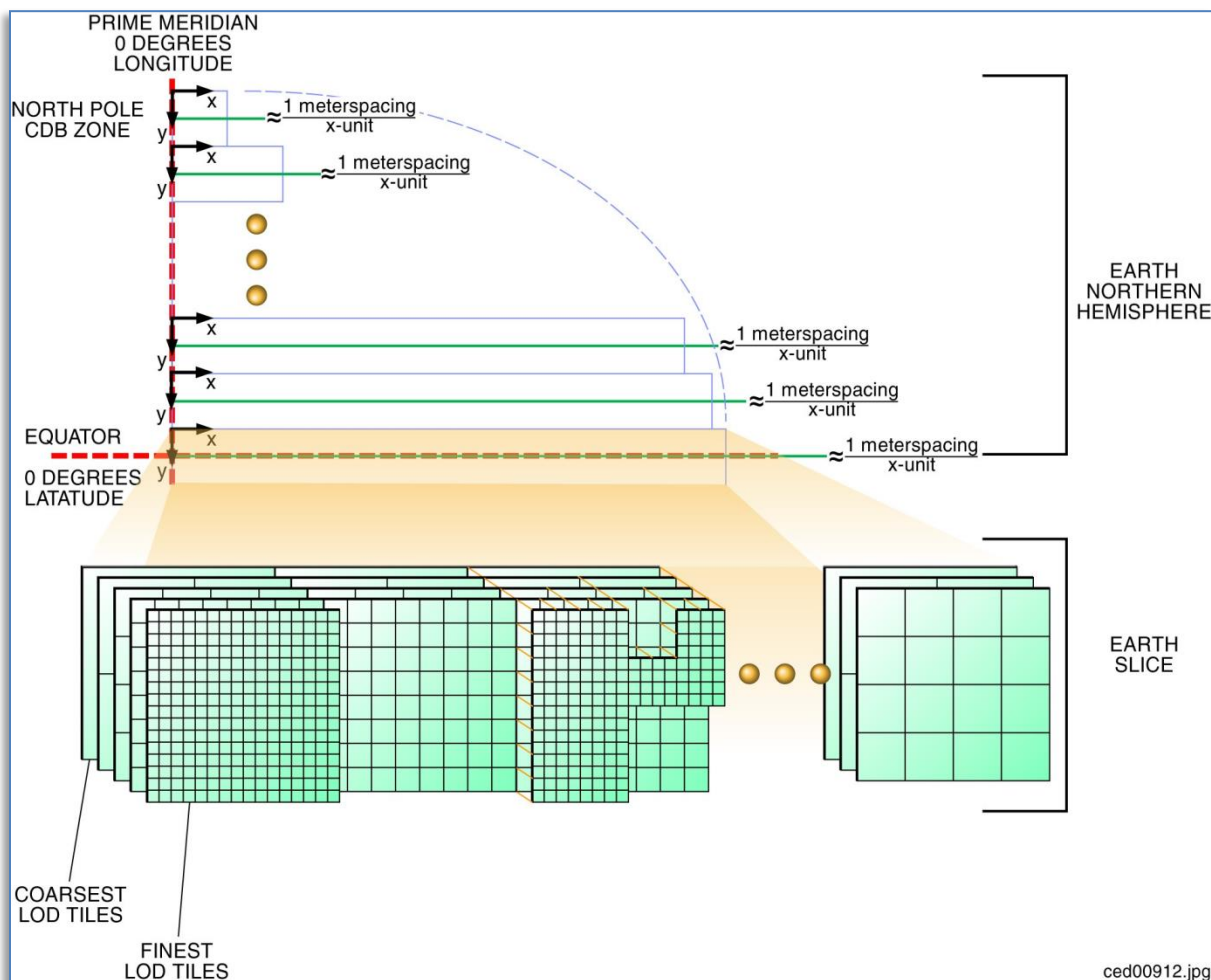


Figure 2-4: Earth Slice Example (Five Levels-of-Detail)

### 2.1.2.2 Tile-LOD Hierarchy Rules

The CDB Specification further imposes that the LOD hierarchy of all tiled dataset must be complete for every CDB Geocell. However, each CDB Geocell may have a different number of Tile-LODs.

### 2.1.2.3 Tile-LOD Replacement Rules

In general, finer tiles replace coarser tiles. The actual rules are:

1. For negative levels of details, a tile at LOD  $n$  replaces exactly one tile at LOD  $n-1$  since both tiles cover the same area.
2. For positive levels of details, a tile at LOD  $n$  is replaced by 4 tiles at LOD  $n+1$  since tiles from LOD  $n+1$  cover only a quarter of the area covered by the tile at LOD  $n$ .

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

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

### 2.1.3 Handling of the North and South Pole

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

As shown in Figure 2-2: Variation of Longitudinal Dimensions versus Latitude, a single CDB Geocell at the poles covers 12 degrees in longitude and 1 degree in latitude within a single slice. As a geographic position gets closer and closer to the poles in terms of latitude, fewer points are required in the data grid; however the CDB Geocell still has a regular rectangular shape. Therefore, this implies that grid points will be progressively super 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  $-90^\circ$  of latitude and  $0^\circ$  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  $+89^\circ$  of latitude and  $0^\circ$  of longitude. The redundant data representing the North Pole found in the other 29 geocells of zone 10 is ignored. The case of raster datasets that make use of the corner grid conventions is an exception since the CDB does not provide the means of representing data at precisely the North Pole ( $+90^\circ$  of latitude and  $0^\circ$  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

The CDB Specification is file system independent, (i.e., it does not mandate the use of a specific file system). However, compliance to the CDB Specification does require that the file system be able to support a minimal set of capabilities listed below:

1. File name/Directory name structure:
  - a. Character set: in accordance to Table 2-5: Character Set Used for CDB Files and Folders.



- b. Length of filename (including path to file): 256 characters or more.
  - c. Length of filename extension: “dot” followed by three characters or more
2. Minimum Directory structure:
  - a. Number of files or directories in root directory: 256 entries or more.
  - b. Number of files or directories per directory (except root): 2048 entries or more
  - c. Depth of directory hierarchy: 9 or more (assuming at least 256 entries per directory level).
  - d. Directory size: 128 KB or more (assuming 64 bytes per directory entry).
3. File Size: 64 Mbytes or more.
4. Number of files per volume: 41,600 files or more (assuming 650 MB CD with 16 Kbyte files.
5. Support for removable media.
6. Support for bootable/non-bootable volume.

## 2.2.1 Character Set

**Table 2-5: Character Set Used for CDB Files and Folders**

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
32	20		64	40	@	96	60	`
33	21	!	65	41	A	97	61	a
<del>34</del>	<del>22</del>	<del>“</del>	66	42	B	98	62	b
<del>35</del>	<del>23</del>	<del>#</del>	67	43	C	99	63	c
36	24	\$	68	44	D	100	64	d
<del>37</del>	<del>25</del>	<del>%</del>	69	45	E	101	65	e
<del>38</del>	<del>26</del>	<del>&amp;</del>	70	46	F	102	66	f
39	27	'	71	47	G	103	67	g
40	28	(	72	48	H	104	68	h
41	29	)	73	49	I	105	69	i
<del>42</del>	<del>2A</del>	<del>*</del>	74	4A	J	106	6A	j
43	2B	+	75	4B	K	107	6B	k
44	2C	,	76	4C	L	108	6C	l
45	2D	-	77	4D	M	109	6D	m
46	2E	.	78	4E	N	110	6E	n
<del>47</del>	<del>2F</del>	<del>/</del>	79	4F	O	111	6F	o
48	30	0	80	50	P	112	70	p
49	31	1	81	51	Q	113	71	q
50	32	2	82	52	R	114	72	r
51	33	3	83	53	S	115	73	s



Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
52	34	4	84	54	T	116	74	t
53	35	5	85	55	U	117	75	u
54	36	6	86	56	V	118	76	v
55	37	7	87	57	W	119	77	w
56	38	8	88	58	X	120	78	x
57	39	9	89	59	Y	121	79	y
<del>58</del>	<del>3A</del>	<del>.</del>	90	5A	Z	122	7A	z
59	3B	;	91	5B	[	123	7B	{
<del>60</del>	<del>3C</del>	<del>,</del>	<del>92</del>	<del>5C</del>	<del>\</del>	<del>124</del>	<del>7C</del>	<del> </del>
61	3D	=	93	5D	]	125	7D	}
<del>62</del>	<del>3E</del>	<del>&gt;</del>	94	5E	^	126	7E	~
<del>63</del>	<del>3F</del>	<del>?</del>	95	5F	_	<del>127</del>	<del>7F</del>	<del>~</del>

### 2.2.2 A word about case-sensitiveness

One of the intent guiding the development of the CDB Specification is the possibility of implementing it on modern operating systems whether they are Windows-like or Unix-like.

Throughout this Specification, the reader will notice that file names 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.

As a result, the CDB Specification demands that all names appear exactly as specified in this document, including the appendices.

## 2.3 Light Naming

The CDB Specification provides the means to unambiguously tag any modeled light point<sup>45</sup> 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 Specification.

<sup>45</sup> The CDB Specification 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.

The CDB Specification 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, lineals, areals, and specialized airport systems
- air, land, and surface platforms
- life forms
- munitions

Each light type defined by this Specification, corresponds to a real-world light type. The Specification provides a definition of each light type, which is representative of the light type's function rather than its characteristics. The client-devices use the light type name as an index to derive the properties and characteristics of the light. The approach is client-device independent because the (device-specific) client rendering parameters are not stored in the CDB and are therefore invisible to the modeler and the database 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 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).

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

**Figure 2-5: Extract from Light Naming Hierarchy**

The name of a light is composed as follows. Starting from the top-most level of the hierarchy, concatenate each of the names encountered with the backslash<sup>46</sup> “\” 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

<sup>46</sup> The CDB Specification 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.



### **2.3.1 Adding Names to the CDB Light Name Hierarchy**

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

First, the modeler or the simulator vendor is required to a) create a new light type name with its corresponding light code, b) provide a definition for the light type name c) insert the new light type into the light name hierarchy d) edit the Lights.xml metadata file to reflect the change to the Light Name Hierarchy and e) 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 Specification.

Since the light type codes are global to the CDB, 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 CDB 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 CDB remain coherent.

## **2.4 Model Component Naming**

The CDB Specification 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 Specification.

The CDB Specification 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 Specification (refer to Appendix F for a listing of the Model Components) corresponds to a real-world model component. 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.

Chapter 6, CDB OpenFlight Models 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 Specification. 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 Specification deals with the handling of materials that make up the synthetic environment. The CDB Specification 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 Specification provides the means to represent:

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

Appendix L of this Specification provides a list of CDB Base Materials. All references to Composite Materials must, in the end, resolve down to one or more of the stated CDB Base Materials.



Sensor simulation typically requires a simulation of the device itself, supplemented by a complete simulation of the synthetic environment over the portion of the electromagnetic spectrum that is relevant to this device. The former simulation is referred to as the Sensor Simulation Model (SSM) while the latter is called the Sensor Environmental Model (SEM). Most SEMs in existence today rely heavily on environmental database whose content is designed to match the functionality, fidelity, structure and format requirements of the SEM. The level of realism possible by the SEM depends heavily on the quality, quantity and completeness of the data available. This makes the environmental database highly device-specific.

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 Specification defines and publicly defines a list of materials that can be used in a CDB. 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. Section A.1 of Appendix A provides a rationale for the approach taken by the CDB Specification.

### **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. The Base Material name must be unique, since it can be used as an index or search key in other tables (described in subsequent sections) of the CDB structure.

**Table 2-6: Components of a Base Material**

<b>Component</b>	<b>Description</b>
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.	

The Base Material's name always begins 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. Appendix L presents all of the Base Materials currently defined by the CDB Specification.



### 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 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.

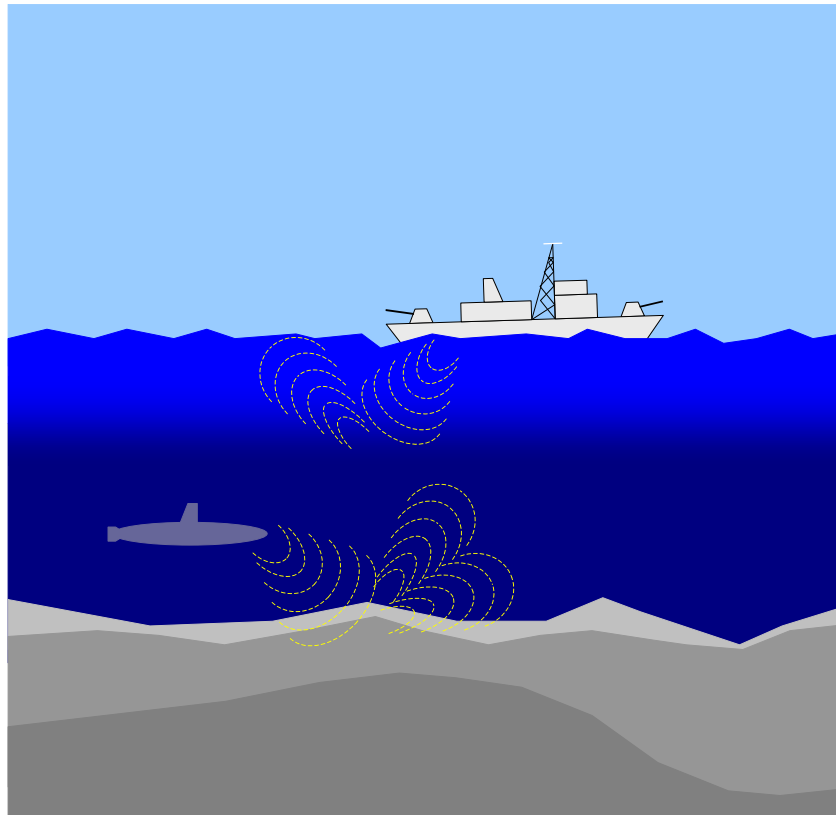
Each Composite Material consists of a primary substrate component, an optional surface component and one or more optional secondary substrate components. Each of these components is in turn composed of one or more Base Materials described in the previous section. Components that are composed of two or 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  $\mu\text{m}$  to 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 must sum to 100%. 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

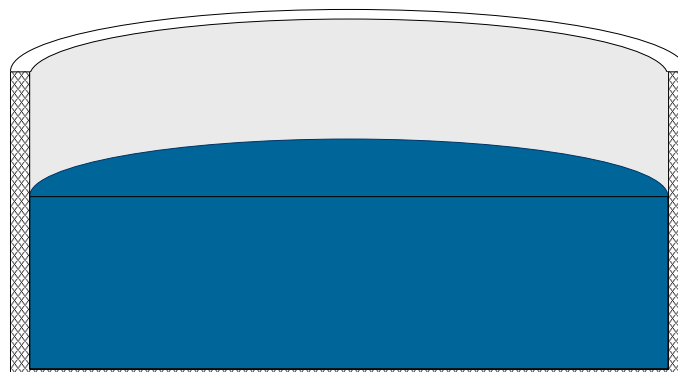
A substrate provides a means to describe the material composition of “hidden” materials located beneath (or inside) the surface of a feature. This information is not explicitly modeled using (for instance) polygons; instead it is an essential characteristic of the material that makes up the modeled feature.

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).

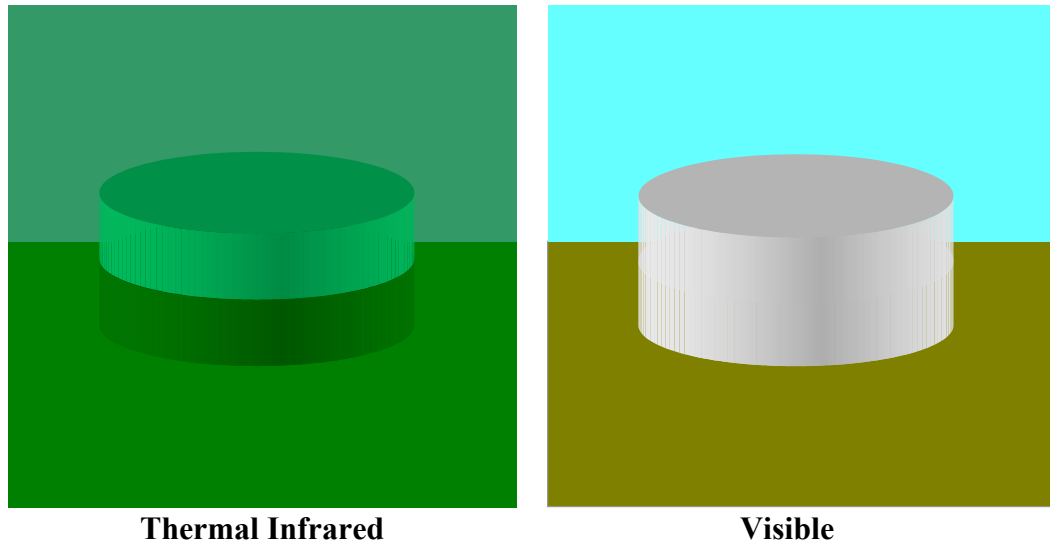


**Figure 2-7: Oil Tank Composite Materials**

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).



**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>
```

There can be only one Primary\_Substrate and it is mandatory. There can be only one Surface\_Substrate and it 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. The base materials that make each substrate must be listed in decreasing order of weighting.

Each composite material must be tagged with a non-negative integer index, zero being reserved for default value that is assigned by the database 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 database tool-specific or modeler-specific composite material name.

### 2.5.2.3 Example 1

Consider a linear feature in the CDB 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 areal feature in the GSFeature dataset of the CDB. The areal 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 areal 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).

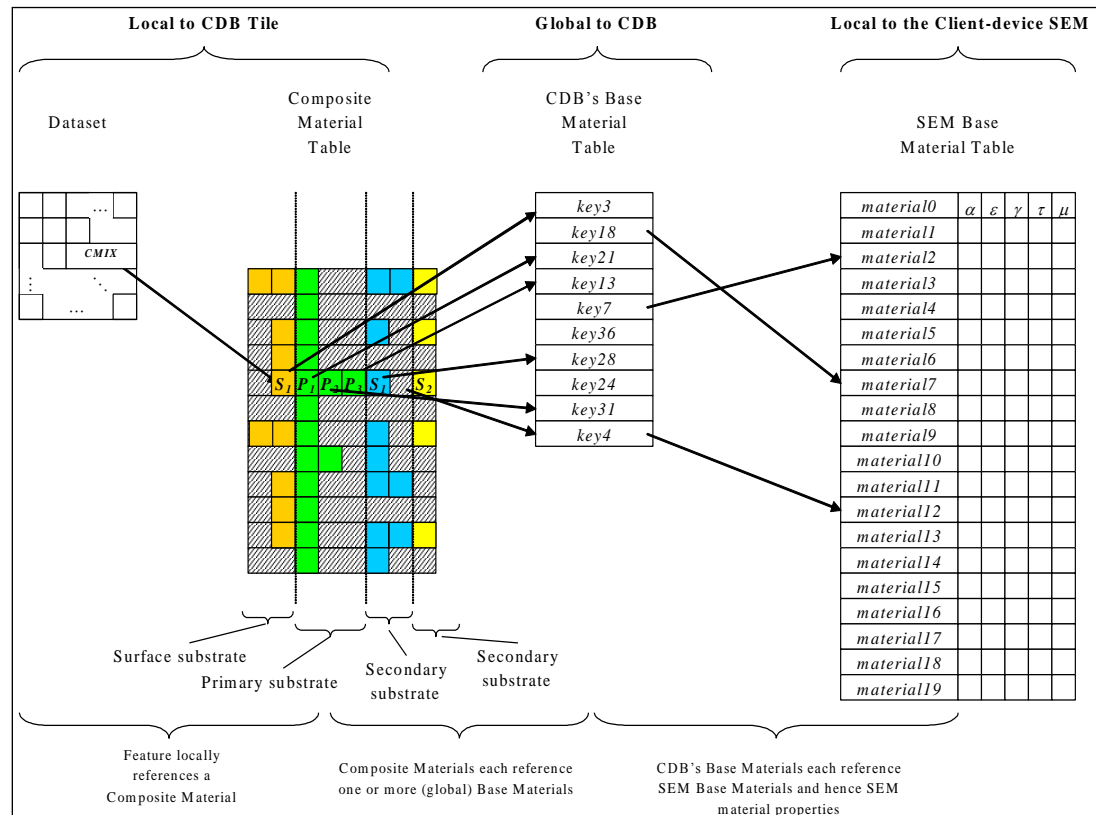
## 2.5.3 Bringing it all Together

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 of the CDB 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 an OpenFlight model 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 Specification.



**Figure 2-9: Flow of Material Attribution Data**

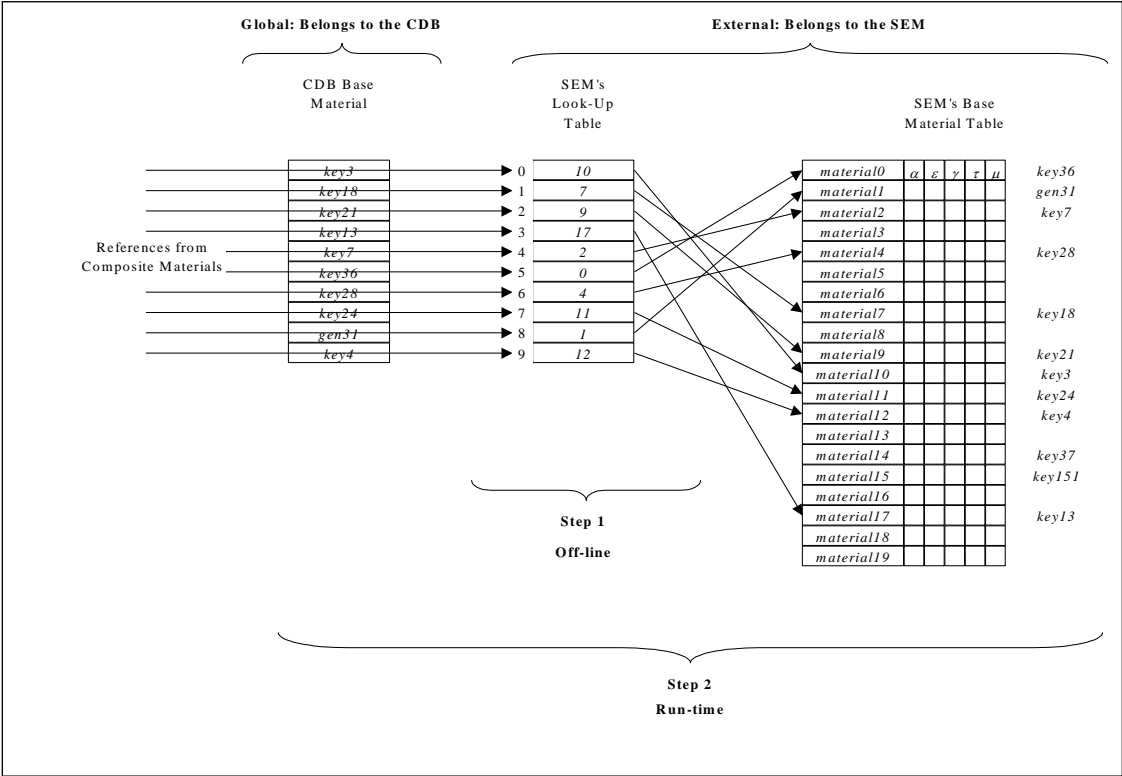
## 2.5.4 Determination of Material Properties by SEM

The association of material properties to features in the CDB requires two distinct steps.

1. The first step consists in establishing a correspondence between all of the Base Materials in the CDB and the Base Materials directly supported by the SEM of the client-device. This is a manual task performed by the SEM specialist(s). The specialist must ensure that his SEM has a corresponding Base Material for each of the CDB Base Materials. In cases where the SEM is simple, it is possible for two or more CDB Base Materials to point to the same SEM Base Material. Alternately the SEM specialist may chose to create new SEM Base Materials that correspond more closely to the CDB's Base Materials. The result of this process is a SEM look-up.
2. The second step is typically undertaken during the CDB initialization by the client-device running the SEM. During this initialization phase, the SEM reads the content of the global Base Material Table and the SEM look-up provided by the SEM specialist. This look-up establishes an indirect link between the Base Materials in the CDB and the material properties of the client-device's SEM Base Materials. In fact, the indirect



link (i.e., the look-up table) can be eliminated if the client device internally builds a Materials Properties Table that uses the CDB material keys directly (as illustrated in Figure 2-11: SEM Base Material Properties Table).



**Figure 2-10: Linking CDB Base Materials to SEM Base Materials**

CDB Base Material Table		SEM's Base Material Properties Table					
References from Composite Materials	key3	0	$\alpha 10$	$\epsilon 10$	$\gamma 10$	$\tau 10$	$\mu 10$
	key18	1	$\alpha 7$	$\epsilon 7$	$\gamma 7$	$\tau 7$	$\mu 7$
	key21	2	$\alpha 9$	$\epsilon 9$	$\gamma 9$	$\tau 9$	$\mu 9$
	key13	3	$\alpha 17$	$\epsilon 17$	$\gamma 17$	$\tau 17$	$\mu 17$
	key7	4	$\alpha 2$	$\epsilon 2$	$\gamma 2$	$\tau 2$	$\mu 2$
	key36	5	$\alpha 0$	$\epsilon 0$	$\gamma 0$	$\tau 0$	$\mu 0$
	key28	6	$\alpha 4$	$\epsilon 4$	$\gamma 4$	$\tau 4$	$\mu 4$
	key24	7	$\alpha 11$	$\epsilon 11$	$\gamma 11$	$\tau 11$	$\mu 11$
	gen31	8	$\alpha 1$	$\epsilon 1$	$\gamma 1$	$\tau 1$	$\mu 1$
	key4	9	$\alpha 12$	$\epsilon 12$	$\gamma 12$	$\tau 12$	$\mu 12$

**Figure 2-11: SEM Base Material Properties Table**

### 2.5.4.1 Example

In the example of Section 2.5.2.4, Example 2, we have a Composite Material consisting of four Base Materials. For the purpose of this example, we will associate hypothetical keys to these materials:

- water (key3 = "BM\_WATER-FRESH", BMT's index 0)
- vegetation (key21 = "BM\_LAND-LOW\_MEADOW", BMT's index 2)
- soil (key7 = "BM\_SOIL ", BMT's index 4)
- sand (key4 = "BM\_SAND ", BMT's index 9)

The SEM specialist establishes the following correspondence between the CDB Base Materials and his materials (step 1):

- key3 to material 8 ("Lake", SEM list's index 8)
- key21 to material 3 ("Uncultivated Land", SEM list's index 3)
- key7 to material 7 ("Soil", SEM list's index 7)
- key4 to material 12 ("Sand", SEM list's index 12)

During the CDB initialization process (step 2), a look-up table is built as follows:

- BMT's index 0 is associated to SEM list's index 8
- BMT's index 2 is associated to SEM list's index 3
- BMT's index 4 is associated to SEM list's index 7
- BMT's index 9 is associated to SEM list's index 12

### 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 areals and/or lineals. Note that the quality of the image classification typically improves with the availability of multi-spectral 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 Appendix L of this Specification. In this case, the modeler need not be aware of the Base Materials mandated by the CDB Specification. 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 Specification and resolve all material references into the Base Materials supported by this Specification. 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 by the CDB.

In the case of 3D models, the modeler is required to 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 Specification and can work with the materials he is most familiar with.



## Chapter 3

### 3 CDB Structure

This chapter defines the structure of the CDB, i.e., the name of all directories forming the CDB hierarchy, as well as the name of all files found in the CDB hierarchy. An important feature of the CDB Specification 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 of all CDB datasets.

#### 3.1 Top-Level CDB 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 Specification 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. All of the files stored within a CDB must be under the root directory or within a subdirectory under the root directory of the CDB, as specified in this document. Run-time applications must be given the path and device on which the CDB is stored in order to access the CDB.

The CDB Specification also has provisions for the handling of multiple, incremental versioning of the CDB. To support this capability, run time applications must first access a predetermined version of the CDB and all its predecessors to determine content changes to the CDB. If no change is encountered in any of the incremental versions, the applications are required to use the content of the active default CDB. The versioning mechanism is done at the file level. Refer to Section 3.2, CDB Configuration Management, for details on how CDB supports incremental versioning.

2. **\CDB\Metadata\:**

The directory that contains the specific XML metadata files which are global to the CDB. The directory structure and metadata descriptions are defined in Section 3.1.1, Metadata Directory.



3. \CDB\GTModel\:

This is the entry directory that contains the Geotypical Models Datasets. The directory structure is as defined in Section 3.4, GTModel Library Datasets.

4. \CDB\MMModel\:

This is the entry directory that contains the Moving Models Datasets. The directory structure is defined in Section 3.5, MModel Library Datasets.

5. \CDB\Tiles\:

This is the entry directory that contains all tiles within the CDB instance. The directory structure is defined in Section 3.6, CDB Tiled Datasets.

6. \CDB\Navigation\:

This is the entry directory that contains the global Navigation datasets. The directory structure is defined in Section 3.7, Navigation Library Dataset

Most of the 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 used throughout the CDB. File content is described in Section 5.1, Metadata Datasets. Most metadata files (except one) are optional and CDB users must implement default behaviors, according to information contained in this Specification. The \CDB\Metadata directory contains the following metadata files:

1. “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 Appendix E. The hierarchy found in Appendix E must be used when “Lights.xml” is not found in the metadata directory. Refer to section 5.1.1 Light Name Hierarchy Metadata for a description of the light point name hierarchy file.

2. “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. A listing of the CDB model components can be found in Appendix F. Refer to section 5.1.2 Model Components Definition Metadata for a description of the model component file.



3. “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 Appendix L. The base material names found in appendix L must be used when the “Materials.xml” file is not found in the metadata directory. Refer to section 5.1.3 Base Material Table for a description of the materials definition file.
4. “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. Appendix S lists the various default values as documented throughout this Specification. Defaults values found in appendix S must be used if the “Defaults.xml” file is not found in the metadata directory. Refer to section 5.1.4 Default Values Definition Metadata for a description of the defaults definition file.
5. “Specification\_Version” Metadata file (deprecated)
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 5.1.6, Version Metadata.
7. “CDB\_Attributes” Metadata file:  
This file is used to describe all the CDB attributes that are supported by the CDB Specification. A complete listing and description of CDB attributes is provided in section 5.7.1.3, CDB Attributes of this Specification. The file is described in section 5.1.7 CDB Attributes Metadata.
8. “Geomatics\_Attributes” Metadata file:  
This file is used to describe all Geomatics attributes that may be referenced by this CDB (refer to section 5.7.1.2.6.2, Geomatics Attributes for a description of Geomatics attributes). Note that the usage of Geomatics attribution falls outside of the jurisdiction of the CDB Specification. Nonetheless, the CDB Specification provides a standardized mechanism to allow users to fully describe each of the Geomatics attributes they wish to insert within the CDB repository structure. The file is described in section 5.1.8, Geomatics Attributes Metadata.
9. “Vendor\_Attributes” Metadata file:  
This file is used to describe all Vendor attributes that may be referenced by this CDB (refer to section 5.7.1.2.6.3, Vendor Attributes for a description of Vendor attributes). Note that the usage of Vendor attribution falls outside of the jurisdiction of the CDB Specification. Nonetheless, the CDB Specification provides a standardized mechanism to allow users to fully describe each of the Vendor attributes they wish to insert within the CDB repository structure. The file is described in section 5.1.9, Vendor Attributes Metadata.

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

#### 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 5.1.10, Configuration Metadata.

### 3.1.2 Metadata File Examples

Each CDB Version has a metadata file whose complete path and file name is:

```
\CDB\Metadata\Version.xml
```

A Forward Looking Infrared client device named “FLIR” has a client specific metadata file having the following directory path and file name:

```
\CDB\Metadata\Lights_FLIR.xml
```

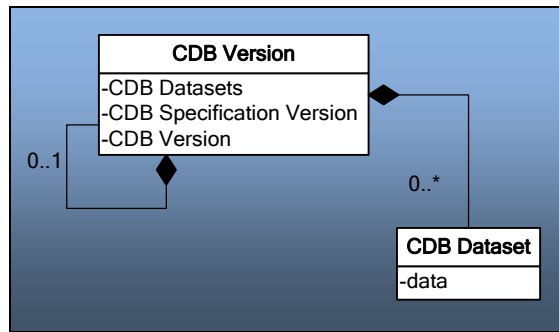
## 3.2 CDB Configuration Management

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

- The Common Database (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

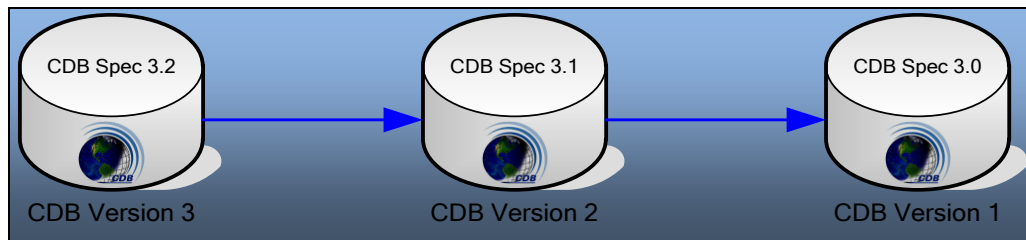
### 3.2.1 CDB Version

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 Specification. 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 Specification 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 Specification 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 Specification. It shows that CDB Version 3 (on the left) complies with version 3.2 of the Specification and refers (the blue line) to CDB Version 2 (in the middle), a 3.1-compliant database, which in turn refers to CDB Version 1 (to the right), a 3.0-compliant database.

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 Specification 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 Specification and is completely left to the user. The following UML diagram defines the CDB Extension concept.

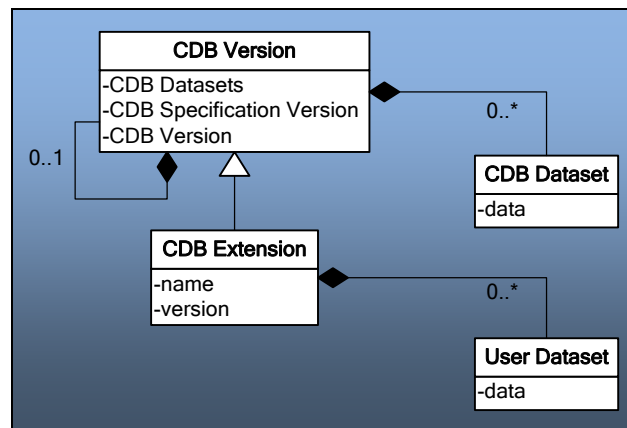


Figure 3-3: UML Diagram of CDB Extension Concept

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.

The root of each CDB Version can have any valid path name<sup>47</sup> on any disk device or volume under the target file system it is stored on. A CDB Version cannot be stored directly in the root directory of a disk device or volume. A CDB Version path name cannot be within another CDB Version. The length of the path name leading to the

<sup>47</sup> As defined in section 2.2, File System

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.

The path to the boot CDB Version is the entry point that must be provided to all client-device applications when loading the CDB synthetic environment. Run-time applications must 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 OpenFlight model. When processed by a client application, the “effective” CDB now contains both the AA051\_Wellhead\_LOD0.flt and the AA051\_Wellhead\_LOD1.flt files.

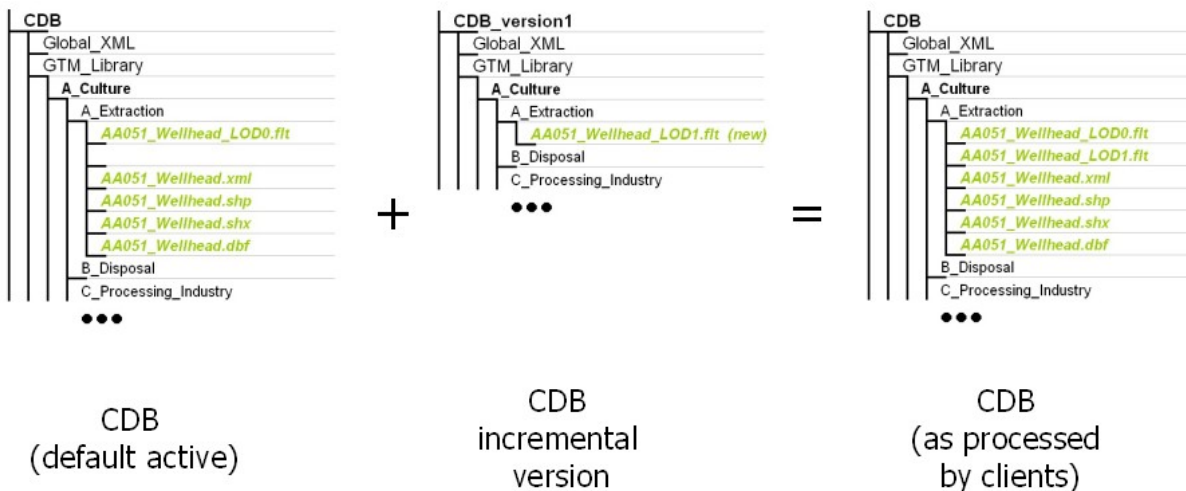
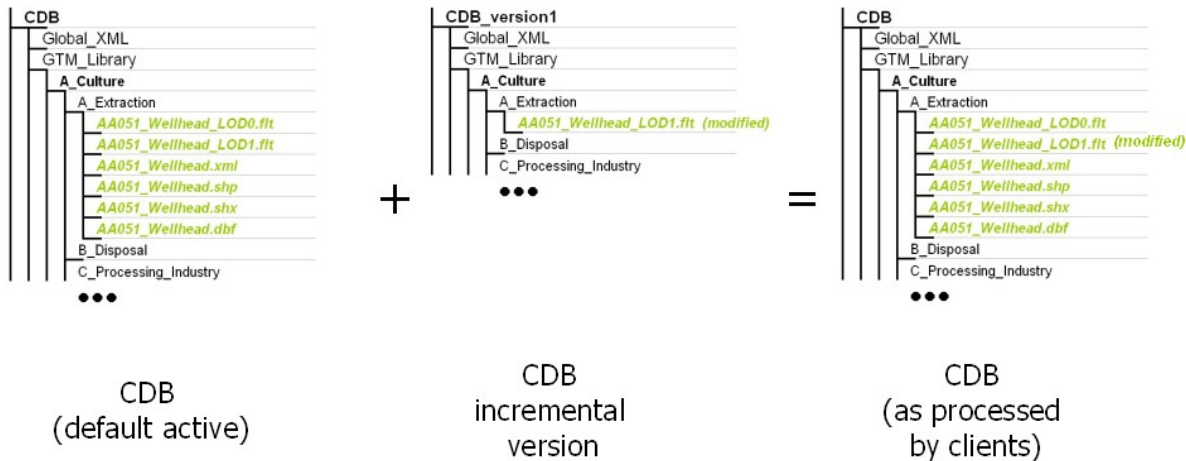


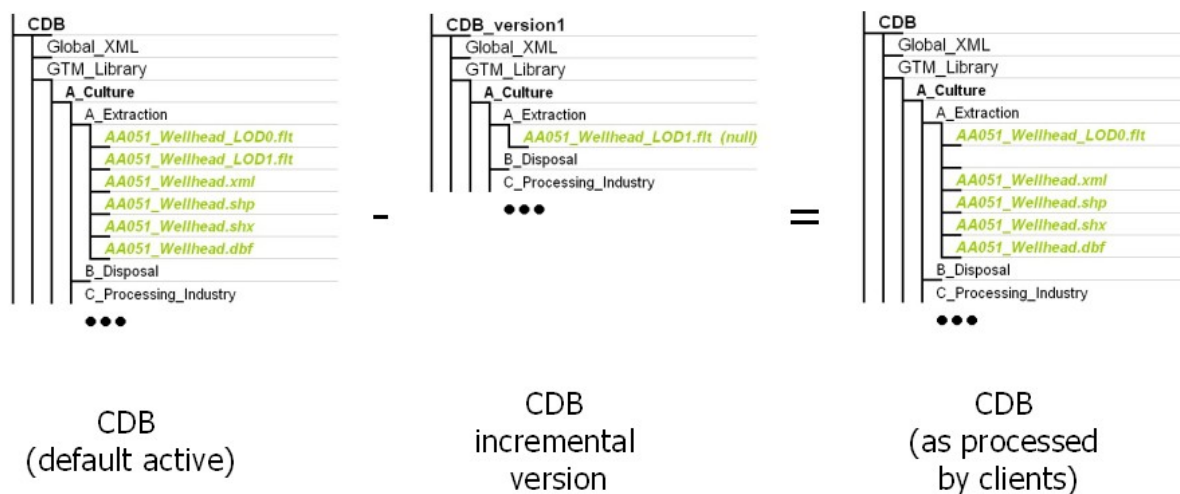
Figure 3-4: Adding content to the CDB

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, illustrates the case where a modeler has modified level-of-detail #1 of a wellhead OpenFlight model. When processed by a client application, the “effective” CDB now contains the modified version of the AA051\_Wellhead\_LOD1.flt.



**Figure 3-5: Modifying Content of the CDB**

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 OpenFlight model. When processed by a client application, the “effective” CDB no longer contains the AA051\_Wellhead\_LOD1.flt OpenFlight file.



**Figure 3-6: Deleting Content from the CDB**



### 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 geospecific 3D model (GSModel) datasets whose files are stored in 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, it must 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.

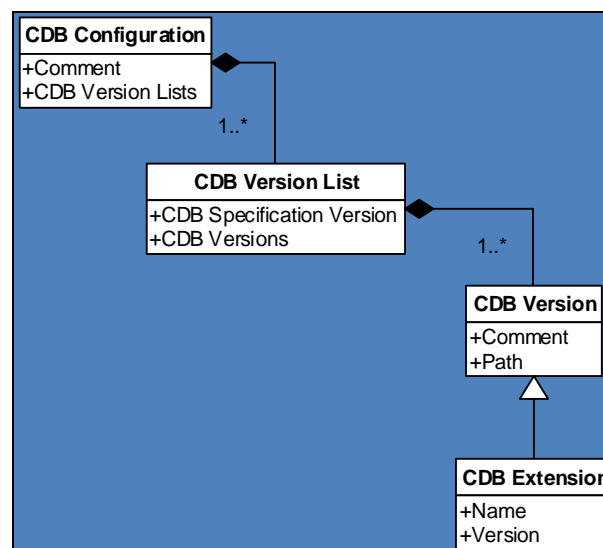


Figure 3-7: UML Diagram of CDB Configuration Concept

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 Specification 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 Specification 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.

Since a CDB is usually intended for use in real-time simulation systems, it is strongly recommended that all CDB chains be limited to no more than 8 CDB versions<sup>48</sup>. 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. Database processing times may increase with chain lengths, commensurate with storage system access times.

### 3.3 CDB Model Types

The cultural features of a CDB can 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.

---

<sup>48</sup> 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).

- b) **GModel**: 3D modeled geospecific representation of a point-, lineal- or areal-feature that is anchored to the ground.
- c) **T2DModel**: 2D modeled geospecific or geotypical representations of point-, lineal and areal 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 Specification, the following terms and expressions will be 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 **GModel**. 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., same FACC) have the same modeled representation.

### 3.3.2 **GModel (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 corresponds to a unique real-world, and

usually 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 Elevation, Imagery, and Raster Material datasets. 2D Models provide the means to represent 2D surface features that are conformed to the underlying terrain:

- a) Modeled representation of geotypical and geospecific 2D lineal-features such as roads, runways and taxiways, stripes.
- b) Modeled representation of geotypical and geospecific 2D areal-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 the CDB Specification, the model is not required to be attached to a cultural point feature.

During the course of a multi-player simulation<sup>49</sup>, 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 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 trade-offs 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

---

<sup>49</sup> 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).

statement as to the expected usage of the feature (and its associated modeled representation) within the CDB.

When a feature is tagged as geotypical...

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

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

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

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

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

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

In most cases, the decision to invoke a modeled representation of a feature as either geotypical or geospecific is clear. When it comes to real-world recognizable cultural features, the representation of these features is clearly a geospecific model because it is encountered once in the entire CDB and it is unique in its shape, texture, etc. At the end of the spectrum, many simulation applications use a generic modeled representation for each feature type and then instance that modeled representation throughout the synthetic environment. For this case, the choice is clearly geotypical.

There are cases however, where the decision to represent features as either geotypical or geospecific is not as clear-cut. For instance, a modeler may not be satisfied with a single modeled representation for all the hospital features (FACC-FSC = AL015-006); accordingly, he may wish to model two or more variants of hospitals in the CDB. While each of these modeled representation may not be real-world specific, they are nonetheless variants of hospitals (say by size or by region or country for example). Usually, the primary motivation for such variations is one of esthetics and realism; it is not necessarily motivated by the need to accurately reflect real-world features.

In making his decision, the modeler should factor-in the following trade-offs:

- a) **CDB Storage Size:** The size of the CDB is smaller when the cultural features reference geotypical models rather than geospecific models. This is due to the fact that the modeled representation of geotypical model is not duplicated within each tile – instead, the model appears once in the GTModel library dataset directory. Clearly, a geotypical model is the preferred choice if the modeler wishes to assign and re-use the same modeled representation for a given feature type.
- b) **Client-device Memory Footprint:** By assigning a geotypical model to a feature, the modeler provides a valuable “clue” to the client-device that the feature will be instanced throughout the CDB with the same modeled representation. As a result, client-device should dedicate physical memory for the storage of the geotypical models for later use.
- c) **GTModel Library Management:** The CDB's Feature Data Dictionary (FDD) is based on the DIGEST, DGIWG, SEDRIS and UHRB geomatics standards. These standards are commonly used for the attribution of source vector data in a broad range of simulation applications. The CDB Feature Data Dictionary acts much like what an English dictionary is to a collection of novels. As a result, it is possible to develop a universal GTModel Library which is totally independent of the CDB content (just like a dictionary is independent of books). This universal GTModel Library can be simply copied into the \CDB\GTModel directory. The structure of the GTModel Library is organized in accordance to the CDB's FDD – in other words, the models are indexed using the CDB Feature Access Code



(FACC). The indexing approach greatly simplifies the management of the model library since every model has a pre-established location in the library.

- d) ***CDB Generation and Update:*** As mentioned earlier, the size of the CDB is smaller when the cultural features reference geotypical models rather than geospecific models. This is due to the fact that the modeled representation of geotypical model is not duplicated within each tile – instead, the model appears once in the GTModel library dataset directory. This reduces the amount of time required by the tools to generate and store the CDB onto the disk storage system. The second benefit of geotypical models comes in the case where a modeler wishes to change the modeled representation of one or more geotypical features type across the entire CDB. Changes to the modeled representation of a feature type can easily be performed by simply overwriting the desired model in model library. From then on, all features of that type now reference the updated model – no other changes to the CBD are required.

Note that since the size of the GTModel library is likely to exceed the client-device's model memory, the client-device must implement a caching scheme which intelligently discards models or portions of models that are deemed less important, used infrequently or not used at all<sup>50</sup>. It is up to the client-device to accommodate for the disparity between the size of client-device's model memory and the size of the GTModel library. Clearly when the disparity is large, the caching algorithm is solicited more frequently and there is more “trashing” of the cache's content. The key to a successful implementation of a caching scheme resides in an approach which discards information not actively or currently used by the client-device. The CDB specification offers a rich repertoire of attribution information so that client-devices can accomplish this task optimally<sup>51</sup>. Consequently, the client-devices can smartly discard model data that is not in use (e.g., models and/or, textures) during the course of a simulation. Note that in more demanding cases, client-devices may have to resort to a greater level of sophistication and determine which levels-of-detail of the model geometry and/or model texture are in use in order to accommodate cache memory constraints. It is clearly in the modeler's interest to avoid widespread usage of model variants within the GTModel Library. In doing so, the modeler overly relies on the client-devices abilities to smartly manage its model cache<sup>52</sup>. As a result, run-time performance may suffer.

---

<sup>50</sup> For example, a caching scheme could be based on the model's frequency of use (computed by the client-device during the simulation), the model's Relative Tactical Importance Code (RTAI), the model's Bounding Sphere (BSR) and scale (SCAL), and the model's Bounding Box (BBH, BBL, BBW). Failure to cache the model would result in the wasteful refetching of the geotypical model from the disk storage system each time it is invoked. This would greatly increase disk storage system IO and reduce client-device paging performance. In the worst-case, the client-device's performance from a memory usage and disk storage bandwidth point-of-view is no worse than of geospecific models

<sup>51</sup> Failure to cache the model would result in the wasteful refetching of the geotypical model from the disk storage system each time it is invoked. This would greatly increase disk storage system IO and reduce client-device paging performance.

<sup>52</sup> Client-devices with less capable memory management might simply refuse to load geotypical models once the memory is exhausted.

As mentioned earlier, the modeled representation of a geotypical model is not duplicated within each tile – instead, the model appears once in the GTModel library dataset directory. As a result, once the model is loaded into memory, it can be referenced without inducing a paging event to the CDB storage system. Clearly, the paging requirements associated with geotypical features are negligible. As a result, paging performance is improved because of the reduced IO requirements on the CDB storage system.

### 3.3.6 Organizing Models into Levels of Details

The geometry, texture, and signature datasets of 3D models are organized into levels of details (LOD) based on their resolutions. The expression of the model resolution depends on the dataset; the resolution of the model geometry is called its Significant Size (SS); the texture resolution is expressed by its Texel Size (TS); and for the radar signature of a model, its resolution is simply its size and is measured by the diameter of its Bounding Sphere (BSD).

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

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

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

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

In all three equations, the value 111319 represents the approximate length in meters of an arc of one degree at the equator<sup>53</sup>.

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**

CDB LOD	ModelGeometry Significant Size	ModelTexture Texel Size	ModelSignature Bounding Sphere
-10	SS > 5.565950 × 10 <sup>+4</sup>	TS > 6.957438 × 10 <sup>+3</sup>	BSD > 4.452760 × 10 <sup>+5</sup>
-9	SS > 2.782975 × 10 <sup>+4</sup>	TS > 3.478719 × 10 <sup>+3</sup>	BSD > 2.226380 × 10 <sup>+5</sup>
-8	SS > 1.391488 × 10 <sup>+4</sup>	TS > 1.739359 × 10 <sup>+3</sup>	BSD > 1.113190 × 10 <sup>+5</sup>
-7	SS > 6.957438 × 10 <sup>+3</sup>	TS > 8.696797 × 10 <sup>+2</sup>	BSD > 5.565950 × 10 <sup>+4</sup>
-6	SS > 3.478719 × 10 <sup>+3</sup>	TS > 4.348398 × 10 <sup>+2</sup>	BSD > 2.782975 × 10 <sup>+4</sup>
-5	SS > 1.739359 × 10 <sup>+3</sup>	TS > 2.174199 × 10 <sup>+2</sup>	BSD > 1.391488 × 10 <sup>+4</sup>
-4	SS > 8.696797 × 10 <sup>+2</sup>	TS > 1.087100 × 10 <sup>+2</sup>	BSD > 6.957438 × 10 <sup>+3</sup>
-3	SS > 4.348398 × 10 <sup>+2</sup>	TS > 5.435498 × 10 <sup>+1</sup>	BSD > 3.478719 × 10 <sup>+3</sup>
-2	SS > 2.174199 × 10 <sup>+2</sup>	TS > 2.717749 × 10 <sup>+1</sup>	BSD > 1.739359 × 10 <sup>+3</sup>

<sup>53</sup> The actual equation to obtain the values of 111319 m is  $L = a \times \frac{\pi}{180^\circ}$  where “a” is the length of the major semi-axis of the WGS-84 ellipsoid; “a” is also known as the equatorial radius.

CDB LOD	ModelGeometry Significant Size	ModelTexture Texel Size	ModelSignature Bounding Sphere
-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 > $8.492966 \times 10^{-1}$	BSD > $5.435498 \times 10^{+1}$
4	SS > $3.397186 \times 10^{+0}$	TS > $4.246483 \times 10^{-1}$	BSD > $2.717749 \times 10^{+1}$
5	SS > $1.698593 \times 10^{+0}$	TS > $2.123241 \times 10^{-1}$	BSD > $1.358875 \times 10^{+1}$
6	SS > $8.492966 \times 10^{-1}$	TS > $1.061621 \times 10^{-1}$	BSD > $6.794373 \times 10^{+0}$
7	SS > $4.246483 \times 10^{-1}$	TS > $5.308104 \times 10^{-2}$	BSD > $3.397186 \times 10^{+0}$
8	SS > $2.123241 \times 10^{-1}$	TS > $2.654052 \times 10^{-2}$	BSD > $1.698593 \times 10^{+0}$
9	SS > $1.061621 \times 10^{-1}$	TS > $1.327026 \times 10^{-2}$	BSD > $8.492966 \times 10^{-1}$
10	SS > $5.308104 \times 10^{-2}$	TS > $6.635129 \times 10^{-3}$	BSD > $4.246483 \times 10^{-1}$
11	SS > $2.654052 \times 10^{-2}$	TS > $3.317565 \times 10^{-3}$	BSD > $2.123241 \times 10^{-1}$
12	SS > $1.327026 \times 10^{-2}$	TS > $1.658782 \times 10^{-3}$	BSD > $1.061621 \times 10^{-1}$
13	SS > $6.635129 \times 10^{-3}$	TS > $8.293912 \times 10^{-4}$	BSD > $5.308104 \times 10^{-2}$
14	SS > $3.317565 \times 10^{-3}$	TS > $4.146956 \times 10^{-4}$	BSD > $2.654052 \times 10^{-2}$
15	SS > $1.658782 \times 10^{-3}$	TS > $2.073478 \times 10^{-4}$	BSD > $1.327026 \times 10^{-2}$
16	SS > $8.293912 \times 10^{-4}$	TS > $1.036739 \times 10^{-4}$	BSD > $6.635129 \times 10^{-3}$
17	SS > $4.146956 \times 10^{-4}$	TS > $5.183695 \times 10^{-5}$	BSD > $3.317565 \times 10^{-3}$
18	SS > $2.073478 \times 10^{-4}$	TS > $2.591847 \times 10^{-5}$	BSD > $1.658782 \times 10^{-3}$
19	SS > $1.036739 \times 10^{-4}$	TS > $1.295924 \times 10^{-5}$	BSD > $8.293912 \times 10^{-4}$
20	SS > $5.183695 \times 10^{-5}$	TS > $6.479619 \times 10^{-6}$	BSD > $4.146956 \times 10^{-4}$
21	SS > $2.591847 \times 10^{-5}$	TS > $3.239809 \times 10^{-6}$	BSD > $2.073478 \times 10^{-4}$
22	SS > $1.295924 \times 10^{-5}$	TS > $1.629905 \times 10^{-6}$	BSD > $1.036739 \times 10^{-4}$
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

GSMModel, 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.

### **3.3.8 Terms and Expressions**

When referring to 3D Models, the Specification 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 Specification has an important Feature Data Dictionary (FDD) whose origins are traceable to the DIGEST v2.1 Specification. However, the current FDD is a consolidation of the DIGEST, DGIWG, SEDRIS, and UHRB dictionaries. The CDB FDD makes use of FACC codes to classify features. To provide an even better classification of features, the Specification defines an additional attribute called the FACC feature sub-code (FSC). By extending the FACC hierarchy structure in this manner, it is possible to define a broader set of model types than is possible with strict implementation of the FACC feature codes. The FACC sub-code value and its significance depend on the FACC feature code. Refer to Appendix N for a list of the FACC feature sub-code.

FACC and FSC are two CDB attributes defined in sections 5.7.1.3.24 and 5.7.1.3.25 respectively.

One of the uses of FACC feature codes by the Specification is to create a hierarchy of subdirectories by taking advantage of the manner in which a FACC is built. A FACC 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 Specification will use these three parts to compose the following hierarchy of folders:

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

Where A is the first character of the FACC code, Category is the category name, B is the second character of the FACC code, Subcategory is the subcategory name, 999 are the 3<sup>rd</sup>, 4<sup>th</sup>, and 5<sup>th</sup> characters of the FACC code, and Type feature type as per Appendix N.

#### **3.3.8.2 Model Name**

When a feature is represented by a 3D model, the model itself is given a name that is used to better describe or differentiate two features having the same FACC and FSC codes. Even though the model name is left to the discretion of the modeler, the Specification recommends the use of the FACC name as the model name. FACC names are listed in appendix N. In the case of Moving Models, the model name is the human-readable version of its DIS Entity Type.

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

### 3.3.8.3 DIS Entity Type

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

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

The first four fields (kind, domain, country and category) are used to create four subdirectories in the moving model datasets hierarchy. Each of the directory names is composed of the field’s value (1 to 3 digits), followed by an underscore “\_”, and concatenated with the field’s name as per Appendix M.

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

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

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

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

### 3.3.8.4 Texture Name

The name of 3D model textures is a character string having a minimum of 2 characters and a maximum length of 32 characters. The first two characters must 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.

The acronym TNAM represents the texture name and is used to compose texture file and directory names. The following directory structure is used by CDB Model texture-related datasets:

```
\A\B\TNAM\
```

The directory represented by \A corresponds to the first character of TNAM in uppercase. The second directory, \B, corresponds to the second character of TNAM

in uppercase. As a result, a texture named “house” will be stored in a directory tree with the following structure:

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

### **3.3.8.5 Level of Detail**

The terms “Level of Detail” and its acronym “LOD” are generally well known to the intended audience of this Specification. In the context of the CDB Specification, filenames and directory names will be composed from the concept of LOD. The Specification uses 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 Specification to designate a particular LOD.

The Specification 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**

The **\CDB\GTModel\** folder is 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 FACC 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 FACC 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 Appendix N.
Level 3	B_Subcategory	The second character of the FACC 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 Appendix N.
Level 4	999_Type	The 3 <sup>rd</sup> , 4 <sup>th</sup> , and 5 <sup>th</sup> characters of the FACC 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 Appendix N.
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

The files of the GTModelGeometry Entry File dataset are stored in level 4 of the 5-level directory structure presented above. The names of the files adhere to the following naming convention:

D500\_Snnn\_Tnnn\_FACC\_FSC\_MODL.flr



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

**Table 3-3: GTModelGeometry Entry File Naming Convention**

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

#### **3.4.1.2 GTModelGeometry Level of Detail Naming Convention**

The files of the GTModelGeometry Level of Detail dataset are stored in level 5 of its directory structure. The names of the files adhere to the following naming convention:

D510\_Snnn\_Tnnn\_LOD\_FACC\_FSC\_MODL.flt

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

**Table 3-4: GTModelGeometry Level of Detail Naming Convention**

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

### 3.4.1.3 GTModelDescriptor Naming Convention

The files of the GTModelDescriptor dataset are stored in level 4 of the 5-level directory structure presented above. The names of the files adhere to the following naming convention:

D503\_Snnn\_Tnnn\_FACC\_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
FACC	Five-character FACC feature code as defined in Appendix N
FSC	Three-digit integer representing the FACC feature sub-code (FSC) as defined in Appendix N
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 FACC 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 FACC categories, \\A\_Culture is the directory containing all FACC subcategories of category A (named Culture), \\L\_Misc\_Feature is the directory containing all FACC types of category A and subcategory L (named Misc\_Feature), \\015\_Building is the directory containing all OpenFlight files representing geotypical buildings whose FACC types are 015 (named Building).

Examples of files found in the above directory are:

.\D500\_S001\_T001\_AL015\_004\_Castle.flt  
.\D500\_S001\_T001\_AL015\_015\_School.flt  
.\D500\_S001\_T001\_AL015\_021\_Garage.flt  
.\D500\_S001\_T001\_AL015\_037\_Fire\_Station.flt  
.\D500\_S001\_T001\_AL015\_050\_Church.flt

Note that all filenames start with a common portion (D500\_S001\_T001\_AL015) and that only their FSC and MODL portions vary.



If the castle above (AL015\_004\_Castle) is represented with 3 levels of details, say LOD 3, 5 and 8, they would be stored in .\L03\, .\L05\, and .\L08\ giving file names such as these:

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

Again, the descriptor associated with the same castle (AL015\_004\_Castle) would be found in this file:

```
.\D503_S001_T001_AL015_004_Castle.xml
```

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

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

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

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

**Table 3-6: GTModelTexture Directory Structure**

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 has from 2 to 32 characters. The first two characters must be alphanumeric.

Note that for compatibility with version 3.0 of the Specification, the name of the directory at level 1 is kept to 501\_GTModelTexture even though dataset 501 has been deprecated and replaced with dataset 511 in version 3.1 of the Specification.

### 3.4.2.1 GTModelTexture Naming Convention

The names of the GTModelTexture files adhere to the following naming convention:

D511\_Snnn\_Tnnn\_LOD\_TNAM.rgb

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

**Table 3-7: GTModelTexture Naming Convention**

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

### 3.4.2.2 GTModelMaterial Naming Convention

The names of the GTModelMaterial files adhere to the following naming convention:

D504\_Snnn\_Tnnn\_LOD\_TNAM.tif

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

**Table 3-8: GTModelMaterial Naming Convention**

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



### 3.4.2.3 GTModelCMT Naming Convention

The names of the GTModelCMT files adhere to the following naming convention:

D505\_Snnn\_Tnnn\_TNAM.xml

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

**Table 3-9: GTModelMaterial Naming Convention**

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

### 3.4.2.4 Examples

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

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

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

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

.\D511\_S001\_T001\_L02\_Brick.rgb  
.\D511\_S001\_T001\_L03\_Brick.rgb  
.\D511\_S001\_T001\_L04\_Brick.rgb  
.\D511\_S001\_T001\_L05\_Brick.rgb  
.\D511\_S001\_T001\_L06\_Brick.rgb  
.\D511\_S001\_T001\_L07\_Brick.rgb  
.\D511\_S001\_T001\_L08\_Brick.rgb  
.\D511\_S001\_T001\_L09\_Brick.rgb  
.\D511\_S001\_T001\_L10\_Brick.rgb



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

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

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

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

```
.\D504_Snnn_Tnnn_LC02_Church-Gothic.tif  
.\D504_Snnn_Tnnn_LC01_Church-Gothic.tif  
.\D504_Snnn_Tnnn_L00_Church-Gothic.tif  
.\D504_Snnn_Tnnn_L01_Church-Gothic.tif  
.\D504_Snnn_Tnnn_L02_Church-Gothic.tif  
.\D504_Snnn_Tnnn_L03_Church-Gothic.tif  
.\D504_Snnn_Tnnn_L04_Church-Gothic.tif  
.\D504_Snnn_Tnnn_L05_Church-Gothic.tif  
.\D504_Snnn_Tnnn_L06_Church-Gothic.tif
```

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

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

### 3.4.3 GTModel Directory Structure 3: Interior Geometry and Descriptor

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

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

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

**Table 3-10: GTModelInteriorGeometry Directory Structure**

<b>Directory Level</b>	<b>Directory Name</b>	<b>Description</b>
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 FACC code (denoted A), called the “Feature Category”, followed by an underscore and the category name (denoted Category) as per Appendix N.
Level 3	B_Subcategory	The second character of the FACC code (denoted B), called the “Feature Subcategory”, followed by an underscore and the subcategory name (denoted Subcategory) as per Appendix N.
Level 4	999_Type	The 3 <sup>rd</sup> , 4 <sup>th</sup> , and 5 <sup>th</sup> characters of the FACC code (denoted 999), called the “Feature Type”, followed by an underscore and the name (denoted Type) associated with the feature type as per Appendix N.
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**

The files of the GTModelInteriorGeometry dataset are stored in level 5 of its directory structure. The names of the files adhere to the following naming convention:

`D506_Snnn_Tnnn_LOD_FACC_FSC_MODL.flr`

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

**Table 3-11: 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.
FACC	Five-character FACC feature code as defined in Appendix N
FSC	Three-digit integer representing the FACC feature sub-code (FSC) as defined in Appendix N
MODL	32-character Model Name String
flt	The file type associated with the dataset (OpenFlight file)

### 3.4.3.2 GTModelInteriorDescriptor Naming Convention

The files of the GTModelInteriorDescriptor dataset are 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\_FACC\_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
FACC	Five-character FACC feature code as defined in Appendix N
FSC	Three-digit integer representing the FACC feature sub-code (FSC) as defined in Appendix N
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 FACC are 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 FACC categories, \A\_Culture is the directory containing all FACC subcategories of category A (named Culture), \L\_Misc\_Feature is the directory containing all FACC types of category A and subcategory L (named Misc\_Feature), \015\_Building is the directory containing all level of details representing geotypical buildings whose FACC types are 015 (named Building), and \L03 is the directory containing the OpenFlight files representing LOD 3 of these buildings.

Examples of files found in the above directory are:

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

Note that all filenames start with a common portion (D506\_S001\_T001\_L03\_AL015) and that only their FSC and MODL portions vary.

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

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

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

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

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

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

**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 must be alphanumeric.

#### 3.4.4.1 GTModelInteriorTexture Naming Convention

The names of the GTModelInteriorTexture files adhere to the following naming convention:

D507\_Snnn\_Tnnn\_LOD\_TNAM.rgb

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

**Table 3-14: GTModelInteriorTexture Naming Convention**

Field	Description
D507	Character D followed by the 3-digit code assigned to the dataset
Snnn	Character S followed by the 3-digit Component Selector 1
Tnnn	Character T followed by the 3-digit Component Selector 2
LOD	The Level of Detail corresponding to the Texel Size of the texture as explained in section 3.3.8.5.
TNAM	The texture name; identical to the folder name where the texture resides.
rgb	The file type associated with the dataset (SGI Image)

#### 3.4.4.2 GTModelInteriorMaterial Naming Convention

The names of the GTModelInteriorMaterial files adhere to the following naming convention:



D509\_Snnn\_Tnnn\_LOD\_TNAM.tif

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

**Table 3-15: GTModelInteriorMaterial Naming Convention**

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

### 3.4.4.3 Example 1

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

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

Where \CDB\GTModel is the root of all geotypical model datasets, \507\_GTModelInteriorTexture is the directory containing all geotypical interior textures, \B is the directory containing all textures whose name start with the letter B, \E is the directory containing all textures whose name start the letter 'B' followed by the letter 'E', and \BeigeGypseWall is the directory containing all LODs of the texture representing a beige gypse wall. Note that the second letter of the texture name is a lowercase 'e' but the corresponding directory name is an uppercase 'E'.

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

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



#### 3.4.4.4 Example 2

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

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

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

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

```
D509_Snnn_Tnnn_L02_Church-Gothic.tif
D509_Snnn_Tnnn_L03_Church-Gothic.tif
D509_Snnn_Tnnn_L04_Church-Gothic.tif
D509_Snnn_Tnnn_L05_Church-Gothic.tif
D509_Snnn_Tnnn_L06_Church-Gothic.tif
D509_Snnn_Tnnn_L07_Church-Gothic.tif
D509_Snnn_Tnnn_L08_Church-Gothic.tif
D509_Snnn_Tnnn_L09_Church-Gothic.tif
D509_Snnn_Tnnn_L10_Church-Gothic.tif
```

#### 3.4.5 GTModel Directory Structure 5: Signature

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

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

The directory structure has 5 levels and is based on the FACC 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.



<b>Directory Level</b>	<b>Directory Name</b>	<b>Description</b>
Level 2	A_Category	The first character of the FACC 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 Appendix N.
Level 3	B_Subcategory	The second character of the FACC 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 Appendix N.
Level 4	999_Type	The 3 <sup>rd</sup> , 4 <sup>th</sup> , and 5 <sup>th</sup> characters of the FACC 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 Appendix N.
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 Specification, 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 Specification.

#### **3.4.5.1 Naming Convention**

The names of the GTModelSignature files adhere to the following naming convention:

D512\_Snnn\_Tnnn\_LOD\_FACC\_FSC\_MODL.xxx

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

**Table 3-17: GTModelSignature Naming Convention**

Field	Description
D512	Character D followed by the 3-digit code assigned to the dataset
Snnn	Character S followed by the 3-digit Component Selector 1
Tnnn	Character T followed by the 3-digit Component Selector 2
LOD	This field is identical to the name of the LOD directory (level 5) where the file is stored.
FACC	Five-character FACC feature code as defined in Appendix N
FSC	Three-digit integer representing the FACC feature sub-code (FSC) as defined in Appendix N
MODL	32-character Model Name String
xxx	The file type associated with the dataset (ESRI Shapefile)

### 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 FACC are 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 FACC categories, \A\_Culture is the directory containing all FACC subcategories of category A (named Culture), \L\_Misc\_Feature is the directory containing all FACC 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 FACC types are 015 (named Building), and \L03 is the directory containing the Shapefiles representing LOD 3 of these buildings.

Examples of files found in the above directory are:

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

### 3.4.6 GTModel Complete Examples

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

```
\CDB\GTModel\500_GTModelGeometry\A_Culture\L_Misc_Feature\
015_Building\
D500_S001_T001_AL015_004_Castle.flt (Entry File)
```



```

D503_S001_T001_AL015_004_Castle.xml           (Descriptor)
Lnn\D510_S001_T001_Lnn_AL015_004_Castle.flt    (LOD)

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

\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 **\CDB\MModel\** folder is the root directory of the MModel library which is 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 Appendix M.
Level 3	9_Domain	The numeric code assigned to the DIS Domain followed by an underscore and the name of the domain as per Appendix 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 Appendix 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 Appendix 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

The names of all MModelGeometry files adhere to the following naming convention:

D600\_Snnn\_Tnnn\_MMDC.flr

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

**Table 3-19: MModelGeometry Naming Convention**

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



### 3.5.1.2 MModelDescriptor Naming Convention

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

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

**Table 3-20: MModelDescriptor Naming Convention**

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

### 3.5.1.3 Examples

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

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

Where \CDB\MModel is the root of all moving model datasets, \600\_MModelGeometry is the directory containing the geometry and descriptor of all moving models, \1\_Platform is the directory containing all DIS Entity of Kind 1 (named Platform), \1\_Land is the directory containing all DIS platforms of Domain 1 (named Land), \225\_United\_States is the directory containing all DIS land platforms of Country 225 (called United\_States), \1\_Tank is the directory containing all DIS land platforms of Category 1 (named Tank), and \1\_1\_225\_1\_1\_8\_0 is the directory containing all geometry and descriptor files of the M1A2 SEP Abrams tank.

Examples of files found in the above directory are:

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

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

This directory structure is owned by the MModelTexture dataset that is assigned dataset code 601. The structure has 4 levels and is based on the texture name (see



section 3.3.8.4). The same directory structure is used to store the files of the MModelMaterial and MModelCMT datasets.

**Table 3-21: MModelTexture Directory Structure**

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 has from 2 to 32 characters. The first two characters must be alphanumeric.

### 3.5.2.1 MModelTexture Naming Convention

The names of all MModelTexture files adhere to the following naming convention:

D601\_Snnn\_Tnnn\_Wnn\_TNAM.rgb

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

**Table 3-22: MModelTexture Naming Convention**

Field	Description
D601	Character D followed by the 3-digit code assigned to the dataset.
Snnn	Character S followed by the 3-digit value of Component Selector 1
Tnnn	Character T followed by the 3-digit value of Component Selector 2
Wnn	Character W followed by the 2-digit Texture Size Code
TNAM	The texture name; identical to directory level 4 above
rgb	The file type associated with the dataset (SGI Image)

### 3.5.2.2 MModelMaterial Naming Convention

The MModelMaterial dataset is assigned dataset code 604 and the names of all MModelMaterial files adhere to the following naming convention:



D604\_Snnn\_Tnnn\_Wnn\_TNAM.tif

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

**Table 3-23: MModelMaterial Naming Convention**

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

### 3.5.2.3 MModelCMT Naming Convention

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

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’, \l 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 Appendix M.
Level 3	9_Domain	The numeric code assigned to the DIS Domain followed by an underscore and the name of the domain as per Appendix 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 Appendix 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 Appendix M.
Level 6	9_9_9_9_9_9_9	All 7 fields of the DIS Entity type concatenated and separated by an underscore.



Directory Level	Directory Name	Description
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

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

```
D606_Snnn_Tnnn_LOD_MMDC.shp  
D606_Snnn_Tnnn_LOD_MMDC.shx  
D606_Snnn_Tnnn_LOD_MMDC.dbf  
D606_Snnn_Tnnn_LOD_MMDC.dbt
```

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
shp shx dbf dbt	The file type associated with the dataset (ESRI Shapefile)

### 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 Shapefiles representing LOD 4 of RCS signature of the tank.

Examples of files found in the above directory are:

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

### 3.5.4 MModel Complete Examples

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

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

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

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

## 3.6 CDB Tiled Datasets

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

### 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 vs Tile and Grid Size. An example of a raster dataset is terrain imagery.

<b>Note:</b>	Partially-filled Tile-LODs are not permitted by the CDB specification. 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) must fill the remainder area of the Tile-LOD with the "best available" data. There are two cases to consider:
<b>Case I:</b>	In the case where coarser LODm data exists for the remainder area of the Tile-LODn, the LODm data should be interpolated to LODn.
<b>Case II:</b>	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 areal 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 <sup>2</sup> )
-10	1	$1.11319 \times 10^{+05}$	$8.06977 \times 10^{-11}$
-9	1	$1.11319 \times 10^{+05}$	$8.06977 \times 10^{-11}$
-8	1	$1.11319 \times 10^{+05}$	$8.06977 \times 10^{-11}$
-7	1	$1.11319 \times 10^{+05}$	$8.06977 \times 10^{-11}$

CDB LOD	Maximum Number of Points per Tile	Approximate Tile Edge Size (meters)	Average Point Density (points/m <sup>2</sup> )
-6	4	$1.11319 \times 10^{+05}$	$3.22791 \times 10^{-10}$
-5	16	$1.11319 \times 10^{+05}$	$1.29116 \times 10^{-09}$
-4	64	$1.11319 \times 10^{+05}$	$5.16466 \times 10^{-09}$
-3	256	$1.11319 \times 10^{+05}$	$2.06586 \times 10^{-08}$
-2	1024	$1.11319 \times 10^{+05}$	$8.26345 \times 10^{-08}$
-1	4096	$1.11319 \times 10^{+05}$	$3.30538 \times 10^{-07}$
0	16384	$1.11319 \times 10^{+05}$	$1.32215 \times 10^{-06}$
1	16384	$5.56595 \times 10^{+04}$	$5.28861 \times 10^{-06}$
2	16384	$2.78298 \times 10^{+04}$	$2.11544 \times 10^{-05}$
3	16384	$1.39149 \times 10^{+04}$	$8.46177 \times 10^{-05}$
4	16384	$6.95744 \times 10^{+03}$	$3.38471 \times 10^{-04}$
5	16384	$3.47872 \times 10^{+03}$	$1.35388 \times 10^{-03}$
6	16384	$1.73936 \times 10^{+03}$	$5.41553 \times 10^{-03}$
7	16384	$8.69680 \times 10^{+02}$	$2.16621 \times 10^{-02}$
8	16384	$4.34840 \times 10^{+02}$	$8.66485 \times 10^{-02}$
9	16384	$2.17420 \times 10^{+02}$	$3.46594 \times 10^{-01}$
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	$8.49297 \times 10^{-01}$	$2.27144 \times 10^{+04}$
18	16384	$4.24648 \times 10^{-01}$	$9.08576 \times 10^{+04}$
19	16384	$2.12324 \times 10^{-01}$	$3.63430 \times 10^{+05}$
20	16384	$1.06162 \times 10^{-01}$	$1.45372 \times 10^{+06}$
21	16384	$5.30810 \times 10^{-02}$	$5.81489 \times 10^{+06}$
22	16384	$2.65405 \times 10^{-02}$	$2.32595 \times 10^{+07}$
23	16384	$1.32703 \times 10^{-02}$	$9.30382 \times 10^{+07}$

For positive LODs, each Tile-LOD of the vector datasets is subject to a limit of 16,384 points to describe the features, whether the file contains point, lineal, or areal features. For negative LODs, this limit is 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. RMDDescriptor
7. GSFeature
8. GTFeature
9. GeoPolitical
10. VectorMaterial
11. RoadNetwork
12. RailRoadNetwork
13. PowerLineNetwork
14. HydrographyNetwork
15. GSModelGeometry
16. GSModelTexture
17. GSModelSignature
18. GSModelDescriptor
19. GSModelMaterial
20. 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 Appendix Q.
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.

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

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

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

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

$$SliceID = \text{int} \left( \frac{lat + 90}{DLatCell} \right)$$
$$NbSliceID = 2 \times \text{int} \left( \frac{90}{DLatCell} \right)$$

where...

*lat* : is the latitude of the CDB tile

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

The CDB Specification sets *DlatCell* to 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 Specification Earth Model Tiled Datasets is  $-90 \leq lat < 90$ ; Refer to Section 2.1.3, Handling of the North and South Pole for the handling of the latitude of +90.

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

### 3.6.2.1.1 Examples

Data elements located at a latitude of  $-5.2^\circ$  will be found under the directory named:

`\CDB\Tiles\S06`

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

`\CDB\Tiles\N62`

### 3.6.2.2 Directory Level 2 (Longitude Directory)

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

The directory name prefix is “E” (East) for longitudes greater than or equal to 0 ( $lon \geq 0$ ) and “W” (West) for longitudes less than 0 ( $lon < 0$ ); three digits follow the prefix:

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

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

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

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

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

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

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 (*DLongCell*) must be determined:

$$DLongCell = DLongCellBasic \times DLongZone(lat)$$

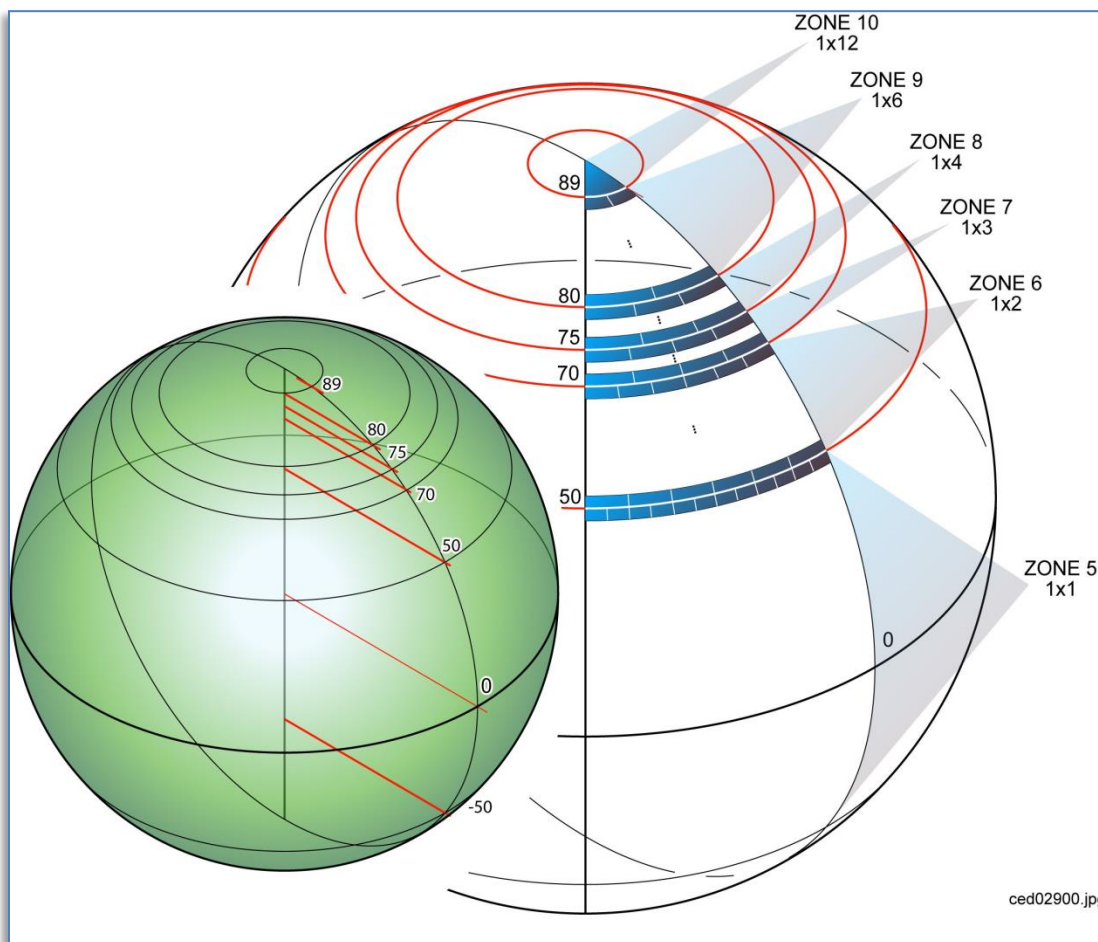
where...

*DLongCellBasic* is the width of a CDB Geocell in degrees at the equator

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

The CDB Specification sets *DLongCellBasic* 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 Specification Earth Model Tiled Datasets is  $-180 \leq lon < 180$  which implies that an application must convert a longitude of 180 to  $-180$  before computing *SliceIDIndex*.

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



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

**Table 3-29: NbSliceIDIndex for every CDB Zones**

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

### 3.6.2.2.1 Examples

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

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

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

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

The reason for “W162” instead of “W161” is that at latitudes between  $50^{\circ}$  and  $70^{\circ}$ , 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 Appendix Q. Examples are provided below.

#### 3.6.2.3.1 Examples

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

```
\CDB\Tiles\S06\E045\001_Elevation  
\CDB\Tiles\S06\E045\004_Imagery
```

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

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

The network of roads covering the geocell located at a latitude of  $+62^{\circ}$  and a longitude of  $-160^{\circ}$  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 of  $-5.2^{\circ}$  and a longitude of  $+45.2^{\circ}$  will be found under the directories named:

```
\CDB\Tiles\S06\E045\300_GSModelGeometry  
\CDB\Tiles\S06\E045\301_GSModelTexture
```

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

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

To complete these examples, the files associated with the Navigation dataset and covering the geocell located at a latitude of  $+36^{\circ}$  and a longitude of  $-88^{\circ}$  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 a latitude of  $-6^\circ$  and a longitude of  $+45^\circ$  will be found under the directory named:

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

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

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

### 3.6.2.5 Directory Level 5 (UREF Directory)

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

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

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:

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

...which simplifies to:

$$U_{N\text{RowLod}} = \max(2^{\text{Lod}}, 1)$$

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

$$D\text{Lat} = (lat + 90) - \text{int} \left( \frac{lat + 90}{D\text{LatCell}} \right) \times D\text{LatCell}$$

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

$$D\text{Lat} = \text{mod}(lat + 90, D\text{LatCell})$$

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



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

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

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

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

### 3.6.2.5.1 Examples

At a latitude of  $-5.2^\circ$  and at LOD 2, the UREF index computed from the above formulas will be:

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

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

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

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

A geospecific feature whose significant size qualifies it for LOD 7 and positioned at a latitude of  $+62.3^\circ$  will produce the following UREF index:

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

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

Assuming a longitude of  $-160.4^\circ$ , 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)

All files stored in the UREF subdirectory of section 3.6.2.5 have the following naming convention:

`LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn.xxx`

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

**Table 3-30: Tiled Dataset File Naming Convention 1**

Field	Description
Lat	Geocell Latitude – Identical to the name of the directory defined in section 3.6.2.1, Directory Level 1 (Latitude Directory).
Lon	Geocell Longitude – Identical to the name of the directory defined in section 3.6.2.2, Directory Level 2 (Longitude Directory).
Dnnn	Character D followed by the 3-digit code assigned to the dataset.
Snnn	Character S followed by the 3-digit value of Component Selector 1.
Tnnn	Character T followed by the 3-digit value of Component Selector 2.
LOD	Level of Detail – As defined in section 3.3.8.5, Level of Detail.
Un	UREF – Identical to the name of the directory as defined in section 3.6.2.5, Directory Level 5 (UREF Directory).
Rn	RREF – A reference to the Right Index of a tile. Character R (Right direction) followed by the column number as described in this section.
xxx	File extension as per file type.

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

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

...which simplifies to:

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

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

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

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

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

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

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

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

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

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

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 a latitude of  $-5.2^\circ$  and a longitude of  $+45.2^\circ$  and at LOD 2, the RREF index computed from the above formulas will be:

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

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

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 a latitude of  $+62.3^\circ$  and a longitude of  $-160.4^\circ$  will produce the following RREF index:

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

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

Resulting 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

These files are stored in archives whose names follow the naming convention defined in section 3.6.3.1 above; the files inside those archives follow the naming conventions defined here.

`LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_"extra_tokens".xxx`

The extra tokens are described in the next sections.

### 3.6.3.2.1 GSModel Geometry File Naming Conventions

The files from the GSModelGeometry and GSModelInteriorGeometry datasets have the following naming convention:

`LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_FACC_FSC_MODL.flr`

The FACC, 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

The files from the GSModelTexture and GSModelInteriorTexture datasets have the following naming convention:

`LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_TNAM.rgb`

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

### 3.6.3.2.3 GSModel Material File Naming Conventions

The files from the GSModelMaterial and GSModelInteriorMaterial datasets have the following naming convention:

`LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_TNAM.tif`

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

### 3.6.3.2.4 GSModel Descriptor File Naming Conventions

The files from the GSModelDescriptor and GSModelInteriorDescriptor datasets have the following naming convention:

`LatLon_Dnnn_Snnn_Tnnn_LOD_Un_Rn_FACC_FSC_MODL.xml`

The FACC, 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 a latitude of +62.3° and a longitude of -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)

Examples of files found inside the above archives are:

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  
N62W162\_D309\_S001\_T001\_L07\_U38\_R102\_AcmeFactory.xml  
N62W162\_D311\_S001\_T001\_L07\_U38\_R102\_AcmeFactory.xml

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

N62W162\_D310\_S001\_T001\_L07\_U38\_R102.flt



### 3.7 Navigation Library Dataset

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

#### 1. NavData

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

#### 3.7.1 NavData Structure

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

**Table 3-31: GTModelGeometry Entry File Directory Structure**

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 Naming Convention

All files of the NavData dataset have the following naming convention:

D400\_Snnn\_Tnnn.dbf

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

**Table 3-32: NavData Naming Convention**

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

##### 3.7.2.1 Examples

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

\CDB\Navigation\400\_NavData\D400\_S001\_T002.dbf







## Chapter 4

### 4 CDB File Formats

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

Furthermore, the translation step into CDB format is typically trivial since the CDB Specification is based on industry-standard native tool formats.

The CDB Specification 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 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.

This chapter concerns itself with the formats used by the CDB. The formats used by the CDB Specification are:

1. **TIFF (\*.tif):** used for the representation of all datasets whose inherent structure reflects that of a two-dimensional regular grid in a Cartesian coordinate system. The primary use of TIFF within a CDB 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 Appendix B of this Specification. It is to be noted that the LZW compression algorithm within the TIFF format is supported and encouraged by the CDB Specification 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 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 Specification provides all of the conventions to geo-reference each geographic dataset. However, it is strongly recommended that database 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 Format (\*.rgb):** used for the representation of 3D model textures. The file format allows for the representation of an image with 1, 2, 3, or 4 channels. A single channel image represents a grey-shaded texture; a two-channel image represents a grey-shaded texture with an alpha component providing the transparency; a three-channel image represents a color (RGB) texture; finally, a four-channel image is a color (RGB) texture with an alpha channel providing the transparency. CDB-compliant RGB readers must be fully compliant with the SGI Image File Format Specification. Its use is limited to 3D models.
4. **JPEG 2000 (\*.jp2):** used for the representation of an image encoded in accordance to the JPEG 2000 standard. CDB-compliant JPEG 2000 readers must be fully compliant with the JPEG 2000 standard while reading such still image file types. JPEG 2000 encoded images can be used for the representation of geo-referenced terrain imagery with some degree of compression levels and is only applicable in the case of terrain raster imagery.
5. **OpenFlight (\*.flt):** used for the representation of 3-dimensional geometric representation of all models (statically positioned cultural models, moving models). To qualify as a CDB-compliant OpenFlight reader, the reader must satisfy the requirements described in Appendix C of this Specification.
6. **Shape (\*.shp, \*.shx, \*.dbf, \*.dbt):** used for the representation and attribution of the vector feature datasets in a CDB. To qualify as a CDB-compliant Shape reader, the reader must satisfy the requirements described in Appendix D of this Specification.
7. **Extensible Mark-up Language (\*.xml):** used to store metadata that describes CDB versioning, describes CDB Composite and Base material structure, defines CDB light type naming conventions and hierarchy, and defines CDB model component hierarchy.
8. **Cross-platform and interoperable file storage and transfer format (\*.zip):** used to archive and store geospecific 3D model datasets. 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.

Appendices B, C, D of this Specification define the required compliancy of CDB readers for the TIFF, OpenFlight and Shape formats. Appendix T describes the JPEG 2000 file format and the last section of Appendix D describes the dBASE III+ file format used by the Shapefile standard. For all other formats (used by this Specification), CDB readers must be fully compliant.

**Table 4-1: CDB File Format Compatibility**

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



## Chapter 5

### 5 CDB Datasets

This chapter provides the description of the content of CDB datasets, except OpenFlight and RCS models that are covered in the following chapters. Chapter 5 also provides the Component Selectors necessary to complete the file names associated with all CDB datasets.

#### 5.1 Metadata Datasets

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

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

**Table 5-1: Component Selectors for Metadata Datasets**

CS1	CS2	File Name
Dataset 700, Metadata		
001	-	Lights.xml
002	-	Model_Components.xml
003	-	Materials.xml
004	-	Defaults.xml
005	-	Specification_Version.xml (Deprecated)
006	-	Version.xml
007	-	CDB_Attributes.xml
008	-	Geomatics_Attributes.xml
009	-	Vendor_Attributes.xml
010	-	Configuration.xml
Dataset 701, Client-Specific Metadata		
-	-	Lights_xxx.xml

For client-specific metadata, the Specification only reserves one dataset code but no component selector. The mechanism is kept for backward compatibility with previous version of the Specification. However, its use is strongly discouraged

because it defeats the very intent of the CDB, which is to promote correlation between client devices by having a single source of data.

### 5.1.1 Light Name Hierarchy Metadata

The light name hierarchy for the CDB is described in detail within the table found in Appendix E of this Specification. For run-time access of this data, clients must be able to retrieve such information. To this end, the Lights Hierarchy Definition metadata is stored in an XML file in the metadata CDB directory as described in Section 3.1.1, Metadata Directories. The name of the file is “**Lights.xml**”.

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

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

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

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

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

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

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



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

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

#### 5.1.1.1 Client Specific Lights Definition Metadata

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

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

*Example:*

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

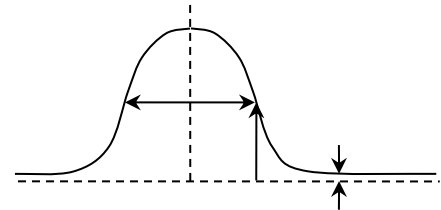
Nominally, the Lights\_xxx.xml consists of light type entries corresponding to the light types the modeler wishes to add/modify. Each entry in the Lights\_xxx.xml file consists of one or optional fields.

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

The XML schema for the fields of the Lights\_xxx.xml is delivered with the Specification in \CDB\Schema\Lights\_Tuning.xsd. The fields are:

- **Intensity:** When a light type is non-native to the CDB specification, which means that it is without a corresponding entry in Appendix E, Intensity represents the light point intensity for the client-device (range normalized from 0.0 to 1.0). When the light entry is native to the CDB specification, Intensity is used as a floating-point intensity modifier that multiplies the intensity calculated by the client-device. In both cases, Intensity defaults to a value of 1.0.
- **Color:** When a light type is non-native to the CDB specification, Color is a floating-point RGB triplet that represents the color of the light type for the client-device (range normalized from 0.0 to 1.0). When the light entry is native to the CDB specification, Color is a floating-point RGB triplet that multiplies the RGB value calculated by the client-device. Color applies only to visual system client-device types. If absent in a light type entry, Color defaults to a value of white (1.0, 1.0, 1.0).

- **Directionality:** A string that categorizes the light type as “Omnidirectional”, “Directional” or “Bidirectional”. If absent in a light type entry, Directionality defaults to the value “Omnidirectional”.
- **Lobe\_Width:** Represents the identifying section for the light’s lobe width characteristics, which can have a horizontal and vertical attribute.
  - **Horizontal:** When a light type is non-native to the CDB specification, the Horizontal field represents the light point’s half-intensity horizontal lobe width for the client-device (range from 0.0 to 360.0). When the light entry is native to the CDB specification, Horizontal field is used as a floating-point modifier that multiplies the horizontal lobe width calculated by the client-device. Applies only to Directional and Bidirectional light types. If absent in a light type entry, Horizontal field defaults to a value of 1.0.
  - **Vertical:** When a light type is non-native to the CDB specification, Vertical field represents the light point’s half-intensity vertical lobe width for the client-device (range from 0.0 to 360.0). When the light entry is native to the CDB specification, Vertical field is used as a floating-point modifier that multiplies the vertical lobe width calculated by the client-device. This applies only to Directional and Bidirectional light types. If absent in a light type entry, Vertical field defaults to a value of 1.0.
- **Residual\_Intensity:** When a light type is non-native to the CDB specification, Residual\_Intensity represents the residual intensity of the light. Residual intensity is the intensity of the light (range normalized from 0.0 to 1.0) outside of the lobe defined by Lobe\_Width:Horizontal and Lobe\_Width:Vertical fields. When the light entry is native to the CDB specification, Residual\_Intensity is used as a floating-point modifier that multiplies the residual intensity calculated by the client-device. This applies only to Directional and Bidirectional light types. If absent in a light type entry, Residual\_Intensity defaults to a value of 1.0.
- **Frequency:** A floating-point value greater than or equal to 0.0 that sets the blink or rotating frequency of the light in Hertz (cycles per second). A value of 0.0 disables all blinking and rotating properties. If absent in a light type entry, Frequency defaults to a value of 0.0.
- **Duty\_Cycle:** A floating-point value ranging from 0.0 to 1.0 that sets the duty cycle of the light. Duty cycle is defined as the percentage of time the light is turned on over a complete cycle. A value of 0.0 permanently turns the light



off. A value of 1.0 turns it on. The value is ignored if Frequency = 0.0. If absent in a light type entry, Duty\_Cycle defaults to a value of 0.5.

Here is a sample of a *Lights\_xxx.xml* file where a modeler has exercised explicit control over the properties of an anti-collision light and a landing light.

```
<Lights_Tuning>
  <Light type="\Light\Platform\Air\Aircraft_Helos\Anti-collision">
    <Description>Tuned for MH-47 CMS</Description>
    <Intensity>0.75</Intensity>
    <Color>1.0 0.0 0.0</Color>
    <Directionality>Omnidirectional</Directionality>
    <Frequency>0.5</Frequency>
    <Duty_Cycle>0.2</Duty_Cycle>
  </Light>
  <Light type="\Light\Platform\Air\Aircraft_Helos\Landing">
    <Description>...</Description>
    <Intensity>1.0</Intensity>
    <Color>1.0 0.9 0.9</Color>
    <Directionality>Directional</Directionality>
    <Residual_Intensity>0.05</Residual_Intensity>
    <Lobe_Width>
      <Horizontal>30.0</Horizontal>
      <Vertical>25.0</Vertical>
    </Lobe_Width>
  </Light>
</Lights_Tuning>
```

### 5.1.2 Model Components Definition Metadata

The CDB specification provides the means to unambiguously tag any portions of a 3D model (moving model or cultural feature with a modeled representation) with a descriptive name. Component model names are stored in the model components definition file, “\CDB\Metadata\Model\_Components.xml” as described in Section 3.1.1, Metadata Directories. Appendix F lists all available model components. The XML schema is provided in \CDB\Metadata\Schema\Model\_Components.xsd delivered with the Specification.

The following shows a content sample of the model component definition file:

```
<Model_Components>

  <Component name="Artillery_Gun">
    <Description>
      1) Refers to any engine used for the discharge of large
        projectiles and served by a crew of men.
      2) Cannon-like weapons operated by more than one person.
    </Description>
  </Component>

  <Component name="Windshield">
    <Description>
      A transparent screen located in front of the occupants of a
      vehicle to protect them from the wind and weather.
    </Description>
  </Component>

  ... other components

</Model_Components>
```

### 5.1.3 Base Material Table

CDB Base Materials are listed in Appendix L and stored in an XML file named \CDB\Metadata\Materials.xml, as mentioned in section 3.1.1. The format of the file is defined by an XML schema that is delivered with the Specification in the file named \CDB\Metadata\Schema\Base\_Material\_Table.xsd.

Here is an excerpt of the CDB Base Material Table showing the definitions of the first and the last base materials of the Specification.

```
<Base_Material_Table>
  <Base_Material>
    <Name>BM_ASH</Name>
    <Description>
      The solid remains of a fire
    </Description>
  </Base_Material>
  ...
  <Base_Material>
    <Name>BM_WOOD-DECIDUOUS</Name>
    <Description>
      Trunks, branches of live deciduous trees
    </Description>
  </Base_Material>
</Base_Material_Table>
```

### 5.1.4 Default Values Definition Metadata

Default values for all datasets can be stored in the default values metadata file “\CDB\Metadata\Defaults.xml” as described in Section 3.1.1, Metadata Directories. Default values defined throughout the CDB specification are listed in Appendix S. The XML schema is provided in \CDB\Metadata\Schema\Defaults.xsd delivered with the Specification. There are two types of default values: read and write default values (‘R’ or ‘W’.) Generally, read default values are values to be used when optional information is not available. Write default values are default values to be used by CDB creation tools to fill mandatory content when information is either missing or not available. The default value name is a unique name identifying a default value for



a given dataset. Valid default value names are listed in Appendix S. Each default value has a type. Valid default value data types are “float”, “integer” and “string”.

The following is an excerpt of a “Defaults.xml” file containing the default terrain elevation value.

```
<Default_Value_Table>
  <Default_Value>
    <Dataset>001_Elevation</Dataset>
    <Name>Default_Elevation-1</Name>
    <Description>Default Primary Terrain Elevation</Description>
    <Type>float</Type>
    <Value>0.0</Value>
    <R_W_Type>R</R_W_Type>
  </Default_Value>

  <!-- Insert other Default Values in accordance to the table above -->
</Default_Value_Table>
```

## 5.1.5 Specification Version Metadata – Deprecated

The content of Specification\_Version.xml has been merged into Version.xml; as such, the use of the file is deprecated as of this version of the Specification.

## 5.1.6 Version Metadata

Each CDB Version requires a version control file that is called Version.xml. Its contents are as follows:

```
<Version>
  <PreviousIncrementalRootDirectory name="Path to another CDB Version"/>
  <Comment>A comment to describe this CDB Version</Comment>
  <Specification version="3.2|3.1|3.0" update="n"/>
  <Extension name="name of the extension" version="version of this extension"/>
</Version>
```

The complete XML schema can be found in \CDB\Metadata\Schema\Version.xsd delivered with the Specification.

The optional <PreviousIncrementalRootDirectory> element is used to refer to another CDB Version. This is the mechanism to use to chain together two CDB versions. The optional <Comment> element is a free-format text to describe the purpose and/or the nature of the data of this CDB Version. The optional <Specification> element indicates the version of the CDB Specification that is used to produce the content of this CDB Version. Note that version numbers of the Specification are limited to the existing versions: 3.2, 3.1, and 3.0. Other values are not permitted. Finally, the optional <Extension> element indicates that this CDB Version is in fact a CDB Extension. Since all elements are optional, a valid CDB Version contains at the very minimum one file, Version.xml, which can be empty.

A version control file that does not have a CDB Extension indicates that the CDB Version holds content that strictly follows the CDB Specification.

A CDB Extension corresponds to user defined information, which is not described or supported by the CDB Specification, stored within the CDB Version. As an example, such additional information could be client or vendor-specific information used to increase system performance. Any user defined information shall not replace or be used in place of existing CDB information. A CDB Extension should only contain vendor or device specific information. CDB content adhering to the CDB specification should only be found in the CDB versions. Client devices not concerned with a CDB extension should ignore all non-CDB compliant content, without loss of information.

### 5.1.7 CDB Attributes Metadata

The CDB attributes are listed and described in section 5.7.1.3 CDB Attributes. The metadata for these attributes is stored in \CDB\Metadata\CDB\_Attributes.xml and the schema can be found in \CDB\Metadata\Schema\Vector\_Attributes.xsd. In essence, the file is the transposition of the text found in section 5.7.1.3 CDB Attributes into a format more appropriate for a computer program.

Its contents are as follows:

```
<Vector_Attributes>
  <Attributes>
    <Attribute>...</Attribute>
    ...
    <Attribute>...</Attribute>
  </Attributes>

  <Units>
    <Unit>...</Unit>
    ...
    <Unit>...</Unit>
  </Units>

  <Scalers>
    <Scaler>...</Scaler>
    ...
    <Scaler>...</Scaler>
  </Scalers>
</Version>
```

The file is composed of three major sections, the first one being the most important. The file has a list of attributes, followed by two lists of units and scalers that are referenced by individual attribute.

#### 5.1.7.1 Definition of the <Attribute> Element

Each attribute is defined as follows:





```
<Attribute code="..." symbol="...">
  <Name>...</Name>
  <Description>...</Description>
  <Level>...</Level>
  <Value>...</Value>
</Attribute>
```

The code is the integer value assigned to each attribute listed in section 5.7.1.3, CDB Attributes. The symbol is the unique character string identifying the attribute. The <Name> is the long form of the symbol. The <Description> is a free form text describing the attribute. The <Level> is defined below and provides the schema level of the attribute. The <Value> element provides the information required to interpret (parse) the value assigned to this attribute.

The schema level is defined as follow:

```
<Level>
  <Instance>...</Instance>
  <Class>...</Class>
  <Extended>...</Extended>
</Level>
```

The <Level> provides a mean to state if the attribute is *Preferred*, *Supported*, *Deprecated*, or *Not Supported* for each of the schema level.

The definition of <Value> is as follow:

```
<Value>
  <Type>...</Type>
  <Format>...</Format>
  <Precision>...</Precision>
  <Range>...</Range>
  <Unit>...</Unit>
  <Scaler>...</Scaler>
</Value>
```

The <Type> is one of *Text*, *Numeric*, or *Boolean*. In the case of a numeric data type, the <Format> indicates if it is a *Floating-Point* or an *Integer* value. For a floating point type, the <Precision> provides the number of digits before and after the decimal point. For numeric types, the <Range> provides the minimum and maximum values; the <Unit> is a reference to a unit code; and the <Scaler> is a reference to a scaler code; both codes being respectively defined in subsequent <Units> and <Scalers> sections.

### 5.1.7.2 Definition of the <Unit> Element

The <Units> section is a list of <Unit> definitions as follow:

```
<Unit code="..." symbol="...">
  <Name>...</Name>
  <Description>...</Description>
</Unit>
```

The code is a positive integer used as a key when a <Value> references a unit. The symbol is the character string that is commonly recognized as the unit identifier. The <Name> is the long form of the unit symbol and <Description> is a free-form text describing this unit.

### 5.1.7.3 Definition of the <Scaler> Element

The <Scalers> section is a list of <Scaler> definitions as follow:

```
<Scaler code="..." symbol="...">  
  <Name>...</Name>  
  <Description>...</Description>  
  <Multiplier>...</Multiplier>  
</Scaler>
```

The code is a positive integer used as a key when a <Value> references a scaler. The symbol is the character string that is commonly recognized as the scaler identifier. The <Name> is the long form of the scaler symbol and <Description> is a free-form text describing this scaler. Finally, <Multiplier> is the numerical multiplier applied to the base unit.

#### 5.1.7.4 Example of CDB\_Attributes.xml

The following example illustrates how to define an attribute:

```
<Vector_Attributes>
  <Attributes>
    <Attribute code="2" symbol="A01">
      <Name>Angle of Orientation</Name>
      <Description>Angle of Orientation with greater than 1 degree resolution.
        The angular distance measured from true north (0 deg) clockwise to the
        major (Y) axis of the feature. If the feature is square, the axis 0
        through 89.999 deg shall be recorded. If the feature is circular, 360.000
        deg shall be recorded.
      </Description>
      <Level>
        <Instance>Preferred</Instance>
        <Extended>Supported</Extended>
      </Level>
      <Value>
        <Type>Numeric</Type>
        <Format>Floating-point</Format>
        <Precision>3.3</Precision>
        <Range interval="Right-Open">
          <Min>0</Min>
          <Max>360</Max>
        </Range>
        <Unit>2</Unit>
      </Value>
    </Attribute>
  </Attributes>
  <Units>
    <Unit code="2" symbol="deg">
      <Name>degree</Name>
      <Description>To mesure an angle</Description>
    </Unit>
  </Units>
  <Scalers>
    <Scaler code="2" symbol="k">
      <Name>kilo</Name>
      <Description>A multiplier: thousand</Description>
      <Multiplier>1000</Multiplier>
    </Scaler>
  </Scalers>
</Vector_Attributes>
```

The schema explains the use of the `interval` attribute of the `<Range>` element.

#### 5.1.8 Geomatics Attributes Metadata

Geomatics attributes (section 5.7.1.2.6.2), are listed in “**Geomatics\_Attributes.xml**” (section 3.1.1). The file uses the `Geomatics_Attributes.xsd` schema.

#### 5.1.9 Vendor Attributes Metadata

Vendor attributes (section 5.7.1.2.6.3), are listed in “**Vendor\_Attributes.xml**” (section 3.1.1). The file uses the `Vendor_Attributes.xsd` schema.

### 5.1.10 Configuration Metadata

The Configuration metadata file provides the means of defining CDB Configurations. The syntax of the file is given below. The complete XML schema is provided in /CDB/Metadata/Schema/Configuration.xsd delivered with the Specification.

```
<Configuration>

  <Comment> An optional comment describing this CDB Configuration. </Comment>

  <Version>
    <Folder path="...">
      <Comment> An optional comment describing this CDB Version. </Comment>
      <Specification version="...">
        <Extension name="..." version="...">
      </Version>

    <!-- Other versions as needed -->

  </Configuration>
```

A <Configuration> is a list of one or more <Version> elements. A <Version> has a mandatory <Folder> element to provide the path to the CDB Version. The other three (3) elements have the same definitions as that of section 5.1.6, Version Metadata.

#### 5.1.10.1 A Note about Folder Path

The use of a relative path to a CDB Version ensures a greater form of interoperability between operating systems and file systems. However, the Specification does not prevent the use of absolute paths<sup>60</sup>. A relative path is expressed relative to the root of the CDB Version containing the Configuration file.

#### 5.1.10.2 Example

Assume that we want to assemble two CDB Versions into a single CDB Configuration. The first CDB Version is located in /CDB/myVersion and has the following Version.xml file.

```
<Version>
  <PreviousIncrementalRootDirectory name="/CDB/theVersion"/>
  <Comment> This is the comment describing myVersion. </Comment>
  <Specification version="3.2"/>
</Version>
```

The second CDB Version complies with version 3.0 of the Specification, is located in /CDB/theVersion, and has the following Version.xml file.

---

<sup>60</sup> On Windows, the path can even be specified using the UNC notation.

```
<Version>
  <Comment> This is the comment describing theVersion. </Comment>
</Version>
```

The resulting CDB Configuration is stored in /CDB/myConfiguration and its Configuration.xml file could look like this:

```
<Configuration>

  <Comment>
    This is an example of a CDB Configuration referring to two CDB Versions.
  </Comment>

  <Version>
    <Folder path="../myVersion"/>
    <Comment> This is the comment describing myVersion. </Comment>
    <Specification version="3.2"/>
  </Version>

  <Version>
    <Folder path="../theVersion"/>
    <Comment> This is the comment describing theVersion. </Comment>
    <Specification version="3.0"/>
  </Version>

</Configuration>
```

Notice the use of relative paths to refer to the CDB Versions. Also notice the addition of the <Specification> element to the second <Version> to explicitly state that it contains data complying with version 3.0 of the Specification.

## 5.2 Navigation Library Datasets

The NavData dataset represents the navigation portion of a CDB. It supports several simulation subsystems such as the Instrument Landing System (ILS), Inertial Navigation/Global Positioning System, and Microwave Landing System Communications. The dataset also provides descriptions of airspaces, airways, heliports, helipads, gates, runways, approaches, and terminals. It also provides information regarding climb procedures out of airports.

The NavData dataset is broken down into a collection of 46 (forty-six) components related to the Flight Navigation. Together, these 46 components combine all of the information currently provided by the following two organizations:

- Navigation System Data Base (produced by Jeppesen) around the ARINC Standard 424-16
- Product Standard for the Digital Aeronautical Flight Information File (DAFIF) produced by the National Intelligence Geospatial Agency

The component selector 2 is set to 001 for basic navigation records and these files are located in the tiled Navigation dataset directories. The component selector 2 is set to 002 for schema files and a value between 101 and 126 for key datasets. Schema and key datasets are located in the global Navigation directory. Component selector 1 and

the file type are as per Table 5-28: Tiled Navigation Dataset. It provides a list of all CDB Navigation components with their designated names and description.

**Table 5-2: Component Selectors for Navigation Dataset**

CS1	CS2	File Extension	Component Name	Component Description
Dataset 400, NavData				
001-046	002	*.dbf	Schema	Lists the data attributes for the given component
	101-126	*.dbf	Key Dataset	Sorted lists used to perform queries within the NavData

- T002: Schema file
- T101: Storage number search key
- T102: Ident search Key
- T103: ICAO search Key
- T104: Frequency search Key
- T106: IATA search Key
- T107: Type search Key
- T108: Additional Ident 1 search Key
- T109: Additional ICAO 1 search Key
- T110: Channel search Key
- T111: Additional Ident 2 search Key
- T114: Range search Key
- T115: Sequence search Key
- T116: Country search Key
- T117: Boundary search Key
- T118: Code search Key
- T120: Additional Ident 3 search Key
- T121: Reserved
- T122: Additional Type 1 search Key
- T123: Additional ICAO 2 search Key
- T126: Additional Ident 4 search Key

The Navigation Dataset uses the Shapefile format. Each of the Navigation features are represented by point features in \*.shp files. Each point feature is matched to a group of attributes (describes in Appendix H) provided by the \*.dbf and, optionally, \*.dbt portions of the Shapefile format. Appendix I provides the enumeration codes for the Navigation data fields.

**Table 5-3: List of Navigation Components**

<b>Component Name</b>	<b>CS1</b>	<b>Shape Type</b>	<b>Component Description</b>
Airport	1	Point	Area or land that is used (or intended for use) for the landing and take-off of aircraft.
AirRefueling	2	Point	A specifically designated airspace where air-to-air refueling operations are normally conducted.
AirRefuelingControl	3	Point	Information regarding the Air Traffic Control Center that controls the airspace within which the refueling track or anchor is located.
AirRefuelingFootnote	4	Point	Supplemental notes defining an Air Refueling component
AirRefuelingPoint	5	Point	Single Point from an Air Refueling structure
AirRefuelingSegment	6	Multipoint	Segment from an Air Refueling structure
AirspaceBoundary	7	Point	Designated airspace within which some or all aircraft may be subject to air traffic control.
AirwayRestriction	8	Point	Altitude and time restrictions for airways, airway segments, or sequences of airway segments
Approach	9	Multipoint	Preplanned instrument flight rule (IFR) for air traffic control approach procedures.
ArrestingGear	10	Point	Safety device consisting of engaging or catching devices, and energy absorption devices for the purpose of arresting both tail hook and/or non-tail hook equipped aircraft
COMMS	11	Point	Voice, radio communications, and facility call sign and frequencies available for same operations between the airport environment and aircraft.
ControlAirspace	12	Multipoint	Sequential listing of vertical and lateral limits, defining airspaces of different classifications, within which air traffic control service is provided
EnrouteAirway	13	Point	A specified route designed for channeling the flow of traffic as necessary for the provision of air traffic services
FirUir	14	Multipoint	Flight Information region - Upper Information Region. Designated airspace within which some or all aircraft may be subject to air traffic control.
Gate	15	Point	Passenger gate at an airport



Component Name	CS1	Shape Type	Component Description
GLS	16	Point	GNSS Landing System
Helipad	17	Line	Designated area usually with a prepared surface used for take-off and landing of helicopters
Heliport	18	Point	Area or land intended to be used for landing and takeoff of helicopters
HoldingPattern	19	Point	Flight path maintained by an aircraft that is awaiting permission to land
ILS	20	Multipoint	Instrument landing system - Precision instrument approach system normally consisting of electronic components and visual aids
Marker	21	Point	Transmitter that radiates vertically a distinctive pattern for providing position information to aircrafts
MilitaryTrainingRoute	22	Point	Routes used by the Department of Defense and associated Reserve and Air Guard Units for the purpose of conducting low altitude navigation and tactical training in both IFR and VFR weather conditions below 10,000 feet MSL at airspeeds in excess of 250 KTS IAS.
MilitaryTrainingRouteAirspace	23	Point	Special use airspace or military operations area associated with a Military Training Route
MilitaryTrainingRouteDescription	24	Point	Supplemental information regarding a Military Training Route
MilitaryTrainingRouteOverlay	25	Multipoint	The width left and right of centerline based on a set of widths at Point Ident and another set of width at the Next Point Ident in one segment record.
MLS	26	Multipoint	Microwave Landing System - precision instrument approach system normally consisting of electronic components and visual aids
MSA	27	Point	Minimum Safe Altitude - altitude below which it is hazardous to fly owing to presence of high ground or other obstacles
Navaid	28	Multipoint	Electronic device on the surface, which provides point-to-point guidance information or position data to aircraft in flight
OffRouteTerrainClrAltitude	29	Polygon	Off-Route Terrain Clearance Altitude - Clearance altitudes in non-mountainous and in mountainous areas
ParachuteJumpArea	30	Point	An area designated for parachute jumping activities.



Component Name	CS1	Shape Type	Component Description
ParachuteJumpAreaBoundary	31	Multipoint	Boundary of a Parachute Jump Area
PathPoint	32	Point	
PreferredRoute	33	Point	A system of routes designed to minimize route changes during the operational phase of flight and to aid in the efficient management of air traffic.
PresetSite	34	Point	Preset Site
RestrictiveAirspace	35	Multipoint	Airspace of defined dimensions identified by an area on the surface of the earth wherein activities must be confined
Runway	36	Line	Rectangular area on a land airport prepared for the landing and takeoff runs of aircraft along its length
SID	37	Multipoint	Standard Instrument Departure - preplanned instrument flight rule (IFR) for air traffic control departure procedure
SpecialUse Airspace	38	Point	Airspace of defined dimensions wherein activities must be confined because of their nature and/or wherein limitations may be imposed upon aircraft operations that are not a part of those activities.
STAR	39	Multipoint	Standard Terminal Arrival - preplanned instrument flight rule (IFR) air traffic control arrival procedure
SupplTerminalData	40	Point	Supplemental terminal data
TerminalProcClimb	41	Point	Terminal Procedure Climb - Min or ATC Climb rates
TerminalProcFeedRoute	42	Multipoint	Terminal Procedure Feeder Route - A route depicted on Instrument Approach Procedures to designate routes for aircraft to proceed from the en route structure to the Initial Approach Fix
TerminalProcMin	43	Point	Terminal Procedure Minima - Height minima data for Terminal Procedure
VFRRoute	44	Multipoint	Preplanned arrival or departure routes for helicopters or light fixed wing aircraft to specified airports or heliports using/in Visual Flight Rules (VFR)
VFRRouteSegment	45	Multipoint	Segment of a VFR Route
Waypoint	46	Point	Predetermined geographical position, used for route or instrument approach definition or progress reporting purposes

## 5.2.1 Schema Files

The schema file lists the data attributes for the given NavData component. It contains the following columns:

**Table 5-4: List of Navigation Schema Attributes**

Attribute	Type	Length	Definition
ShortName	String	11	A null-terminated string (ten characters or less). Short-hand name of the attribute used in the tiled ShapeFiles (the dBASE III+ .dbf format limits the field names to 10 characters or less)
DataType	String	255	The data type for the attribute
KeyId	Int	4	Index key for the attribute, used when performing a query. Not all attributes have an assigned index key, as only a few attributes can be used to perform a query. For each attribute with an index key, an index key dataset will be created.

For schema files, the value of CS2 shall be T002.

Each attribute with an index Key (KeyId) has an index key dataset created. The index key dataset includes the last three characters of the KeyId inside the component selector 2 (ex. KeyId 2101 would be dataset component selector 2 – T101).

### 5.2.1.1 Example

Here is the data content of the schema file for the Airport dataset (D400\_S001\_T002.dbf):

**Table 5-5: Example of a Navigation Schema**

ShortName	DataType	KeyId
StoraNumbe	UInt64	2101
AHGT	Logical	
AlterNam	String	
AsCoStNumb	UInt64	
BeacoAvail	Logical	
City	String	
CivMilTyp	CivilMilitaryType	
ClearStatu	ClearanceStatus	
Country	CountryEntry	2116
DayliTim	Float32	
DayTimFram	String	



ShortName	Data Type	KeyId
FlipPage	String	
FuelType	String	
HydElePres	Logical	
IataCode	String	
IcaoCode	String	2103
Ident	String	2102
IfrCapab	Logical	
IslanGrou	String	
Jasu	String	
LonRunLeng	UInt32	
LonRunSurf	PavementType	
MagTruIndi	MagneticTrueIndication	
MagneVaria	Float32	
MgrsPosit	String	
Name	String	
NavIcaCod	String	
NavaiIden	String	
Notam	NotamSystem	
OilType	String	
OperaAgenc	String	
OperaHour	OperatingHours	
Point1	GeoCoordinate	
Remark	String	
ServiRemar	String	
SpeedLimit	UInt32	
SpeLimAlti	Sint32	
StateName	StateEntry	
SupFluTyp	String	
TerraImpac	Logical	
Timezone	Float32	
TransAltit	Sint32	
TransLeve	Sint32	

As per this example, four Airport attributes can be used to perform queries:

- StoraNumbe (key index 2101)
- Ident (key index 2102)
- IcaoCode (key index 2103)
- Country (key index 2116)

## 5.2.2 Key Datasets

The index Key Datasets are sorted lists used to perform queries within the NavData. For each attribute that has an index key in the schema file, an index Key Dataset must be created. For Key Datasets, the Dataset Component Selector 2 shall include the last three digits of the index key from the schema file.

**Table 5-6: List of Navigation Key Attributes**

Attribute	Type	Length	Definition
Value	String	255	Value of the data attribute sorted in increasing order (numbers or characters)
Lat ID	Signed Integer	3	Latitude index of the Geocell which contains the data record
Lon ID	Signed Integer	4	Longitude index of the Geocell which contains the data record
Row ID	Integer	4	Index of the data record in the Geocell starting at 1.

This information can then be used to rapidly lookup which CDB Tile contain the data in the pageable NAV dataset (401) and use the Object ID to access the data record in this dataset.

The Storage number is a Primary Surrogate key that uniquely identifies each record within each NAV dataset sub components.

### 5.2.2.1 Example

For the Airport NavData Component, there shall be 4 key datasets (for attributes StoraNumber, Ident, IcaoCode and Country):

```

\CDB\Navigation\400_NavData\D400_S001_T101.dbf
  (StoraNumber, key index 2101)
\CDB\Navigation\400_NavData\D400_S001_T102.dbf
  (Ident, key index 2102)
\CDB\Navigation\400_NavData\D400_S001_T103.dbf
  (IcaoCode, key index 2103)
\CDB\Navigation\400_NavData\D400_S001_T116.dbf
  (Country, key index 2116)

```

The following is an excerpt from the D400\_S001\_T102.dbf file (Key Dataset for the Ident attribute):

**Table 5-7: Example of Navigation Keys**

Value	Object ID	Lon ID	Lat ID
00CA	2	-117	35
00UT	3	-113	37
00WI	6	-90	44



Value	Object ID	Lon ID	Lat ID
01LS	4	-92	30
01MT	3	-115	48
01WI	2	-91	44
02P	0	-78	40
03AZ	5	-111	31
03CO	3	-105	40
03GA	5	-84	31
04CA	10	-118	34
04MS	4	-91	32
04NV	1	-116	35
05CL	2	-123	38
05LS	2	-93	31
05UT	0	-111	37
06FA	0	-81	26
06MN	1	-93	47
06MO	7	-95	39
06TE	10	-96	30
07FA	7	-81	25
07MT	1	-107	48

For example, the Airport with Ident 04CA shall be found in the Geocell with southwest corner at N34:00:00/W118:00:00. It will be the 10th record in the corresponding Shapefile.

Here is an example of the Storage number being used as a reference between Navigation types:

- Type: Approach
- Attributions:
  - StoraNumbe → Storage number (Approach)
  - AirStoNumb → Airport storage number (referenced)

In this case, we see the Approach navigation type referencing the Airport navigation type by using the Airport Storage number.

## 5.3 CDB Model Textures

The following table provides the Component Selectors associated with all kinds of textures that are usable on geotypical (GT), geospecific (GS), moving (MM), and tiled (T2D) models.

In the context of CDB model textures, the first component selector is known as the “Texture Kind” and the second component selector is simply called the “Texture

Index”. Column 1 lists all texture kinds supported by the Specification. The second column gives the range of indices allowed for each kind.

**Table 5-8: Component Selectors for CDB Model Textures**

CS1 (Kind)	CS2 (Index)	Component Name	Component Description
001	001	Year-Round Texture	Base textures for year-round usage on model shells or general base textures for model interiors.
002	001..012	Monthly Texture	Base textures for monthly usage on the shell of models (enumeration values in Appendix O, details in section 6.13.5.2)
003	001..004	Seasonal Texture	Deprecated – Replaced with kind 009
004	001..999	Uniform Paint Scheme	Base textures for Moving Models with Uniform Paint Schemes (enumeration values in Appendix O, details in section 6.13.5.2)
005	001..999	Camouflage Paint Scheme	Base textures for Moving Models with Camouflage Paint Schemes (enumeration values in Appendix O, details in section 6.13.5.2)
006	001..999	Airline Paint Scheme	Base textures for Moving Models with Airline Paint Schemes (enumeration values in Appendix O, details in section 6.13.5.2)
007	001..999	Shadow Map	Base textures of Moving Models Shadows to be projected onto terrain and/or culture (details in section 6.13.5.1.2)
008	001..999	Motion Blur Texture	Base textures for use with rotating parts (details in section 6.9.2.3)
009	001..004	Quarterly Texture	Base textures for quarterly usage on the shell of models (enumeration values in Appendix O, details in section 6.13.5.2)
051	001..999	Night Map	Subordinate textures to simulate the effect of lights inside 3D model shells (details in section 6.13.5.3)
052	001..999	Tangent-Space Normal Map	Subordinate textures used to simulate the effect of irregular surfaces (details in section 6.13.5.5)
053	001..999	Light Map	Subordinate textures to simulate the effect of lights on surrounding surfaces (detail in section 6.13.5.4)
054	001..999	Contaminant	Subordinate textures to represent the presence of particules on runways, taxiways, and roads in general (enumeration values in Appendix O, details in section 6.13.5.7)
055	001..999	Skid Mark	Subordinate textures to represent the visible mark left by any solid which moves against another one; especially marks of tires on roads and runways (enumeration values in Appendix O, details in section 6.13.5.7)
056	001..999	Detail Texture	Subordinate texture used to add detail to the surface. In most cases, modelers use detail textures to add a finer scaled texture to the base texture (details in section 6.13.5.6)
057	001..999	Cubic Reflection Map	Subordinate textures to simulate reflective surfaces (details in section 6.13.5.8)
058	001..999	Gloss Map	Subordinate textures providing the glossiness of a surface on a per-pixel basis (details in section 6.13.5.9)
099	001	Night Map	Deprecated – Replaced with kind 051
	002	Bump Map	Deprecated – Replaced with kind 052
	003	Light Map	Deprecated – Replaced with kind 053





Appendix O enumerates all textures allocated to kind 002, 003, 004, 005, 006, and 055.

## 5.4 GTModel Library Datasets

Table 5-9 provides the component selector values associated with all GTModel datasets.

**Table 5-9: Component Selectors for GTModel Datasets**

CS1	CS2	File Extension	Component Name	Component Description
Dataset 500, GTModelGeometry				
001	001	*.flt	Geometry Entry File	OpenFlight files containing the references to both the shell and interiors of all levels of detail of geotypical models.
Dataset 510, GTModelGeometry				
001	001	*.flt	Geometry Level of Detail	OpenFlight files containing the geometry of the shell of geotypical models for a given level of detail.
Dataset 506, GTModelInteriorGeometry				
001	001..999	*.flt	Interior Geometry	OpenFlight files describing the geometry of the interior of geotypical models for a given level of detail. The value of Component Selector 2 is the file number. Multiple files are used when the complexity of the interior justifies using more than one file.
Dataset 503, GTModelDescriptor Dataset 508, GTModelInteriorDescriptor				
001	001	.xml	Descriptor	Provides the metadata associated with a GTModel. See section 6.14, Metadata, for a description of the content.  <b>NOTE:</b> A model descriptor includes a Composite Material Table for the exclusive use by its corresponding model geometry datasets above. This CMT is not to be confused with the GTModelCMT and GTModelInteriorCMT datasets below.
Dataset 511, GTModelTexture Dataset 507, GTModelInteriorTexture				
-	-	*.rgb	Texture	Individual base and subordinate textures applied on model geometry, see the complete list in section 5.3, CDB Model Textures.
Dataset 504, GTModelMaterial Dataset 509, GTModelInteriorMaterial				
001	001..255	*.tif	Composite Material Index	Each texel is an index into the associated Composite Material Table (dataset 505 and 513 below). CS2 is the layer number.

CS1	CS2	File Extension	Component Name	Component Description
002	001..254	*.tif	Composite Material Mixture	Each texel indicates the proportion (between 0.0 and 1.0) of the composite material found in the corresponding material layer. Component Selector 2 is the layer number. When the texels are of integral types, they are scaled to the range 0.0 to 1.0.
Dataset 505, GTModelCMT Dataset 513, GTModelInteriorCMT				
001	001	*.xml	Composite Material Table	The Composite Material Table is associated with Material Textures; it contains the definition of the composite materials referenced by the model material datasets above. Its format is as specified in section 2.5.2.2, Composite Material Tables (CMT)
Dataset 512, GTModelSignature (See notes 1 and 2 below)				
001..999	001..016	*.shp *.shx *.dbf	RCS Signature	The Shapefile containing the Radar Cross Section of a geotypical model as described in Chapter 7.
	017..032	*.dbf	RCS Class Attributes	The class-level attributes associated with the RCS Signature file as described in Chapter 7.

Note 1: For GTModelSignature dataset, CS1 refers to the “RCS Frequency” and is used to indicate at which frequency (in MegaHertz) the dataset was generated for. The value of CS1 represents a power of 10 of the frequency and ranges from 1 to 999. The range of frequencies that can be represented is from  $10^1$  MHz to  $10^{999}$  MHz.

Note 2: For GTModelSignature dataset, CS2 refers to the “RCS Polarization Type” and is used to indicate how the electromagnetic field is polarized at transmission and reception by typical Radar. The value can range from 1 to 16 for the instanced-level attributes and from 17 to 32 for the class-level attributes.



Polarization Type (CS2)		Description
Instance Attribute	Class Attribute	
1	17	LINEAR Polarization
2	18	CIRCULAR Polarization
3	19	ELLIPTICAL Polarization
4	20	SINGLE HH Polarization
5	21	SINGLE HV Polarization
6	22	SINGLE VV Polarization
7	23	SINGLE VH Polarization
8	24	DUAL HH-HV Polarization
9	25	DUAL VV-VH Polarization
10	26	DUAL HH-VV Polarization
11	27	ALTERNATING HH-HV Polarization
12	28	ALTERNATING VV-VH Polarization
13	29	POLARIMETRIC HH Polarization
14	30	POLARIMETRIC VV Polarization
15	31	POLARIMETRIC HV Polarization
16	32	POLARIMETRIC VH Polarization

## 5.5 MModel Library Datasets

Table 5-10 provides provide the component selector values associated with all Mmodel datasets.

**Table 5-10: Component Selectors for Mmodel Datasets**

CS1	CS2	File Extension	Component Name	Component Description
Dataset 600, MmodelGeometry (See note 1 below)				
001..999	001..999	*.flt	Geometry	OpenFlight files containing the geometry of Mmodels as described in Chapter 6.
Dataset 601, MmodelTexture				
-	-	*.rgb	Texture	Individual base and subordinate textures applied on model geometry, see the complete list in section 5.3, CDB Model Textures.
Dataset 603, MmodelDescriptor				
001	001	*.xml	Descriptor	Provides the metadata associated with a Mmodel. See section 6.14, Metadata, for a description of the content.  <b>NOTE:</b> The MmodelDescriptor includes a Composite Material Table for the exclusive use by the MmodelGeometry dataset. This CMT is not to be confused with the MModelCMT dataset below.
Dataset 604, MmodelMaterial				

CS1	CS2	File Extension	Component Name	Component Description
001	001..255	*.tif	Composite Material Index	Each texel is an index into the Composite Material Table (dataset 605). Component selector 2 is the layer number.
002	001..254	*.tif	Composite Material Mixture	Each texel indicates the proportion (between 0.0 and 1.0) of the composite material found in the corresponding material layer. Component Selector 2 is the layer number. When the texels are of integral types, they are scaled to the range 0.0 to 1.0.
Dataset 605, MModelCMT				
001	001	*.xml	Composite Material Table	This is the composite material table for use with MmodelMaterial dataset. Its content is described in section 2.5.2.2, Composite Material Tables (CMT).
Dataset 606, MmodelSignature (See notes 2 and 3 below)				
001..999	001..016	*.shp *.shx *.dbf	RCS Signature	The Shapefile containing the Radar Cross Section of a moving model as described in Chapter 7.
	017..032	*.dbf	RCS Class Attributes	The class-level attributes associated with the RCS Signature file as described in Chapter 7.

Note 1: For the MmodelGeometry dataset, the geometry of a moving model can be made of one or more parts, each stored in one or more files depending on how complex a part is.

The value of CS1 represents the part number. A Moving Model has at least one part, the model itself that is also used as a master file for all the other parts when applicable. This is part number 1. Other parts are numbered sequentially. An example of an extra part is a removable external fuel tank.

The value of CS2 is the file number. It is used when the complexity of a part requires using more than one file. The file number starts with 1. A part that references external files does it through OpenFlight Xref nodes.

Note 2: For MmodelSignature dataset, CS1 refers to the “RCS Frequency” and is used to indicate the range of frequencies (in MegaHertz) the dataset was generated for. The range is the set of frequencies from  $10^{CS1-1}$  MHz without exceeding  $10^{CS1}$  MHz.

Note 3: For MmodelSignature datasets, CS2 refers to the “RCS Polarization Type” and is used to indicate how the electromagnetic field is polarized at transmission and reception by typical Radar. The value can range from 1 to 16 for the instance-level attributes of polarizations and from 17 to 32 for the class-level attributes of polarizations.



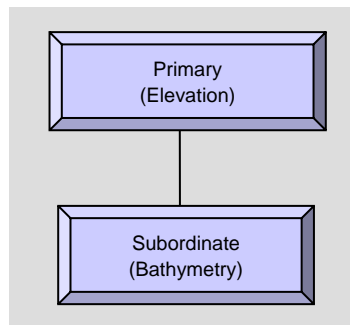
Polarization Type (CS2)		Description
Instance Attribute	Class Attribute	
1	17	LINEAR Polarization
2	18	CIRCULAR Polarization
3	19	ELLIPTICAL Polarization
4	20	SINGLE HH Polarization
5	21	SINGLE HV Polarization
6	22	SINGLE VV Polarization
7	23	SINGLE VH Polarization
8	24	DUAL HH-HV Polarization
9	25	DUAL VV-VH Polarization
10	26	DUAL HH-VV Polarization
11	27	ALTERNATING HH-HV Polarization
12	28	ALTERNATING VV-VH Polarization
13	29	POLARIMETRIC HH Polarization
14	30	POLARIMETRIC VV Polarization
15	31	POLARIMETRIC HV Polarization
16	32	POLARIMETRIC VH Polarization

## 5.6 Tiled Raster Datasets

A raster dataset consist in an evenly spaced grid of data elements that are positioned (in geographic units) along the north-south and east-west axis. This section describes all of the CDB raster datasets.

Most of the CDB raster datasets are broken down further into components. A component is a specialization of the dataset. For example, bathymetry is a specialization of altimetry data because it is targeted to the representation of submerged terrain surfaces; the bathymetric depth data represents altimetry (e.g., heights) with respect to the Primary Elevation component. Together, the Primary Elevation and Bathymetry components form the Elevation Dataset.

A component can be either 1) “primary”, (i.e., it can be used on a stand-alone basis) or 2) “subordinate”, (i.e., it must be used in conjunction with one or more primary components and one or more subordinate components). Subordinate components depend on information contained within another component. Subordinate components are used to progressively add complexity and/or information to a primary component or to another subordinate component. For instance, the Elevation component is a primary component that contains information to allow a simulator client-device to accurately represent the terrain profile or determine the terrain height. On the other hand, Bathymetry is a subordinate component because it cannot be used stand-alone and that it is implicitly subordinate to the Elevation component. It uses the Elevation component to determine the depth of an ocean.

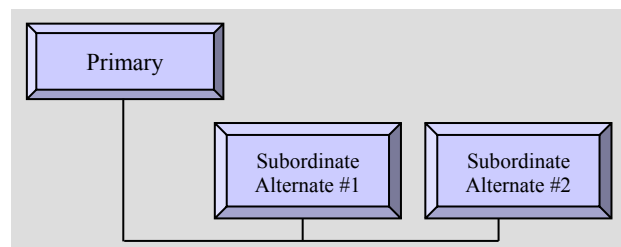


In addition to the notion of primary and subordinate, the CDB embodies the notion of Component Alternates. A component is said to be an alternate component if it can be used interchangeably with other components.

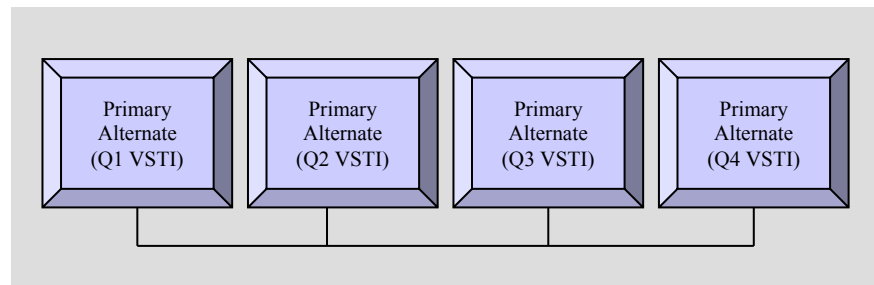
The two concepts can be used in combination as follows:

1. Primary
2. Subordinate or
3. Primary Alternate
4. Subordinate Alternate

A component is Primary if the component is not dependent on another component, i.e., it can be used on a stand-alone basis. Conversely, a component is said to be Subordinate if the dataset is dependent on another component, be it a primary component or another subordinate component. For example, the Bathymetric component is referenced to the CDB Primary terrain elevation component; as a result, it is a subordinate component. The Primary elevation component is a primary component because it does not depend on any other component.



A component is Primary Alternate if a) the component is not dependent on another component, be it a primary component or another subordinate component and b) other primary components can be used interchangeably with the component. For example, the VSTI Q1, Q2, Q3 and Q4 components are all primary alternate components, because they each form the primary layer of terrain imagery, yet they can be used interchangeably.



Finally, a component is Subordinate Alternate if a) if the component is dependent on another component, be it a primary component or another subordinate component and b) other subordinate components can be used interchangeably with the component.

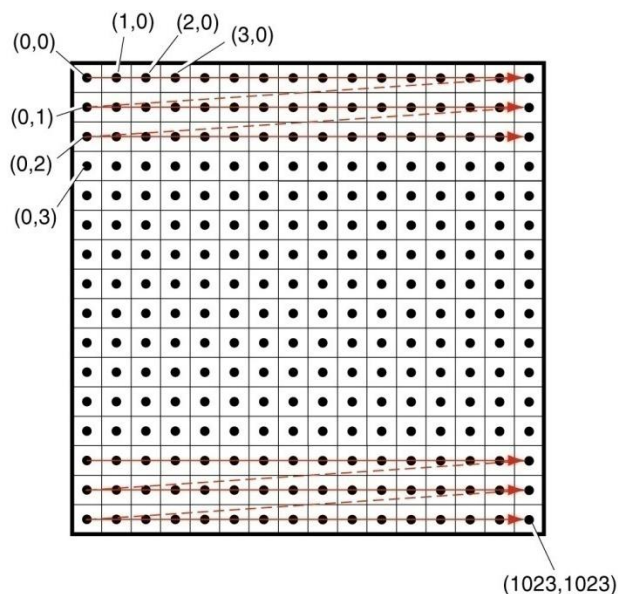
In the case of alternate components, one of them is designated as the default component; in the event that an alternate component is missing in the CDB, the client-devices are required to revert to the default alternate component.

Since subordinate components usually improve the overall fidelity of the dataset, client-devices can revert to the primary component in the event that a subordinate component is missing in the CDB. This behavior allows the CDB Specification to meet one important objective which is to allow any simulator client-device with relatively low performance to still be able to run a CDB implementation (scalability).

Conceptually, the raster dataset tile's internal grid structure uniformly subdivides the tile in both axes. The main characteristic of raster tile is that the number of data elements and the position of every data element are implicit. The application must derive the data element position from the geodetic tile position. The tile grid structure is aligned to the tile edge boundary. The grid elements are organized in row, column order, starting from the northwest corner scanning towards the southeast. This is true for tiles in both the south and north hemisphere. The CDB Specification accommodates for data elements that can be aligned either to the centers or to the corners of the internal tile grid structure. In both cases, the number of data elements in the tile is a power of two. Furthermore, data elements can either represent values representative of samples on the earth surface (e.g., altitude at a point) or values representative of a surface area on the earth surface (average altitude over square area).

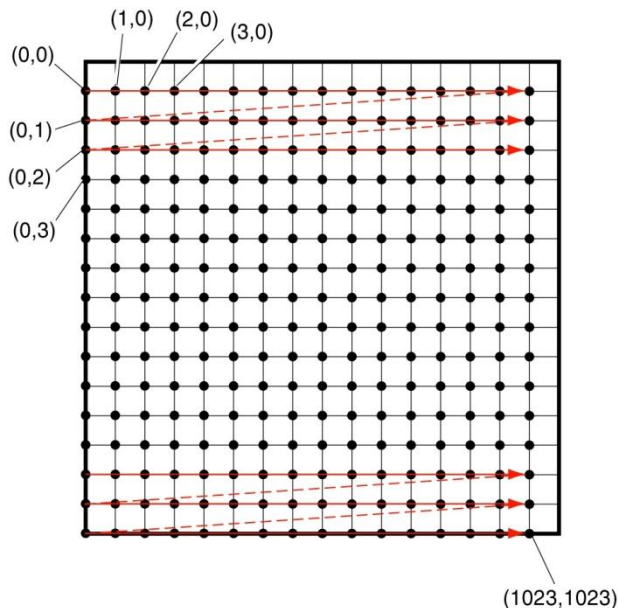
Figure 5-1: Center Grid Data Elements, illustrates a CDB tile with a grid of “center grid data elements” overlaid onto the tile's grid structure with the addressing conventions and the alignment of the samples and areas assigned to each of the data elements.





**Figure 5-1: Center Grid Data Elements**

Figure 5-2: Corner Grid Data Elements, illustrates a CDB tile with a grid of “corner grid data elements” overlaid onto the tile’s grid structure with the addressing conventions and the alignment of the samples and areas corresponding to the data elements.



**Figure 5-2: Corner Grid Data Elements**

Figure 5-3: Center Grid Data Elements as a Function of LODs, illustrates an implementation of center grid data elements with four levels-of-details. Note the shift in position of the data element centers along the x- and y-axis as we shift to

progressively coarser levels-of-detail. Note also that the edges of the data element areas stay aligned with x- and y-axis as we shift to progressively coarser levels-of-detail.

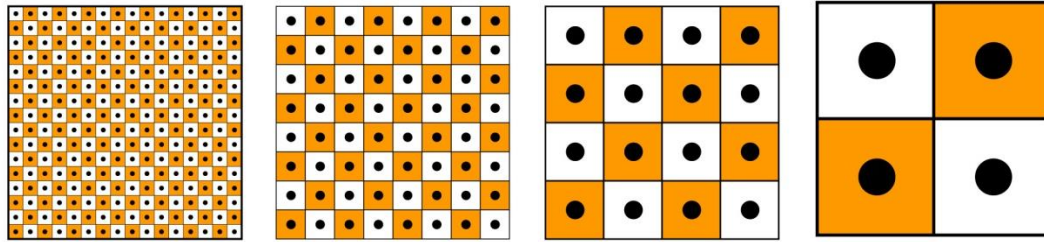


Figure 5-3: Center Grid Data Elements as a Function of LODs

Figure 5-4: Corner Grid Data Elements as a Function of LODs, illustrates an implementation of corner grid data elements with four levels-of-details. Note the shift in the edge of the data element area along the x- and y-axis as we shift to progressively coarser levels-of-detail. Note also that the position of the data elements areas stay aligned with x- and y-axis as we shift to progressively coarser levels-of-detail.

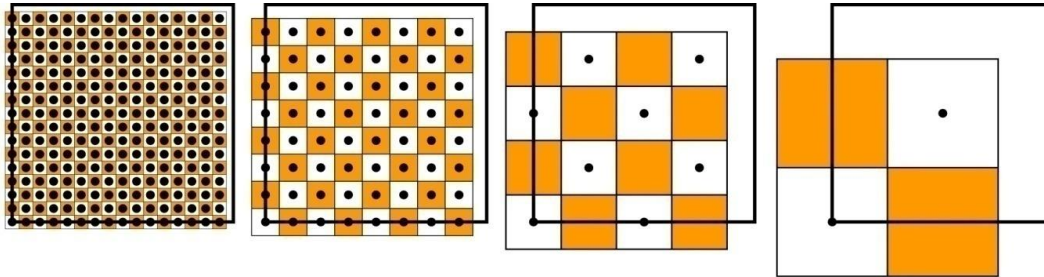


Figure 5-4: Corner Grid Data Elements as a Function of LODs

Sections 5.6.1, 5.6.2, and 5.6.3 describe all of the raster datasets, namely the Tiled Elevation Dataset, the Tiled Imagery Dataset, and the Tiled Raster Material Dataset; each of these sections describes the associated “corner” versus “center” conventions. This convention is intrinsic to the corresponding dataset and is not parametrical. Any changes to these implicit properties require an additional specific dataset to ensure compatibility with applications.

The latitude and longitude of an implicit corner data element (in a tile) with coordinates  $(i, j)$  is computed as per the following equation (note the equation for  $Xunit_{LOD}$  and  $Yunit_{LOD}$  can be found in eq. 3-4)

$$\begin{aligned} Latitude &= TileLatitude - (j \times Yunit_{LOD}) \\ Longitude &= TileLongitude + (i \times Xunit_{LOD}) \end{aligned} \quad (eq. 5-1)$$

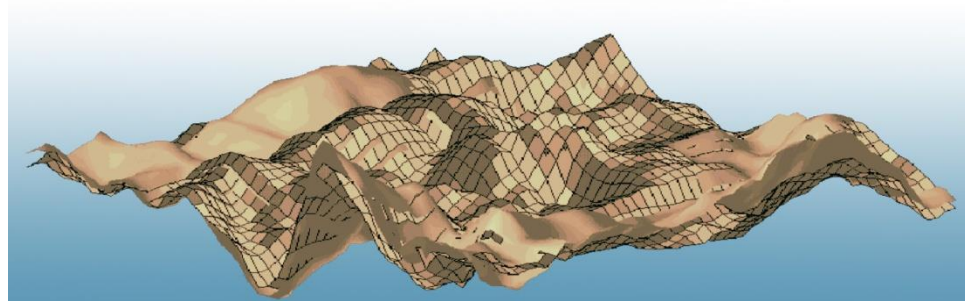
Where  $TileLatitude$ ,  $TileLongitude$  are the tile coordinates at the upper left corner and  $(i, j)$  are the data element coordinates.

Similarly, the position of an implicit center data element in a tile is computed as per the following equation:

$$\begin{aligned} \text{Latitude} &= \text{TileLatitude} - ((j + 0.5) \times YUnit_{LOD}) \\ \text{Longitude} &= \text{TileLongitude} + ((i + 0.5) \times XUnit_{LOD}) \end{aligned} \quad (\text{eq. 5-2})$$

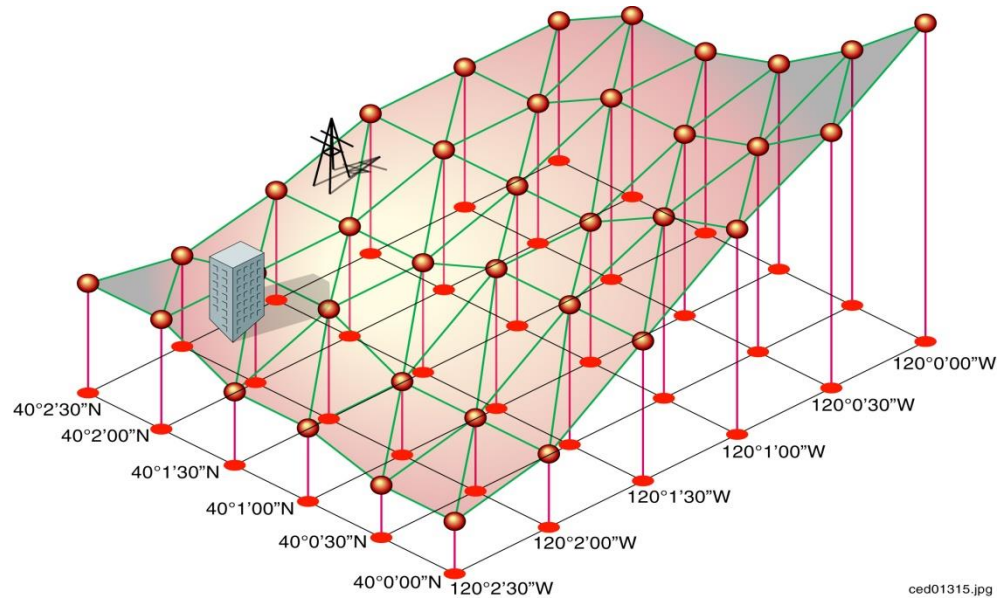
### 5.6.1 Tiled Elevation Dataset

In a CDB, terrain elevation is depicted by a grid of data elements at regular geographic intervals and at prescribed locations within the tile; each grid element is associated with an elevation value. The resultant is a Digital Elevation Model (DEM) of the earth surface with respect (above or below) to the WGS-84 reference ellipsoid. The Elevation Dataset implicitly follows the corner grid element conventions.



**Figure 5-5: Example of Digital Elevation Model (DEM)**

The  $(x, y)$  coordinates of each grid element are its longitude and latitude, respectively. The Elevation dataset holds the vertical extent of the terrain. In Figure 5-6: DEM Depicted as a Grid of Elevations at Regular Sample Points, obstacles such as a tower and a building have been overlaid on the terrain grid to demonstrate that obstacle heights are relative to the terrain height.



**Figure 5-6: DEM Depicted as a Grid of Elevations at Regular Sample Points**

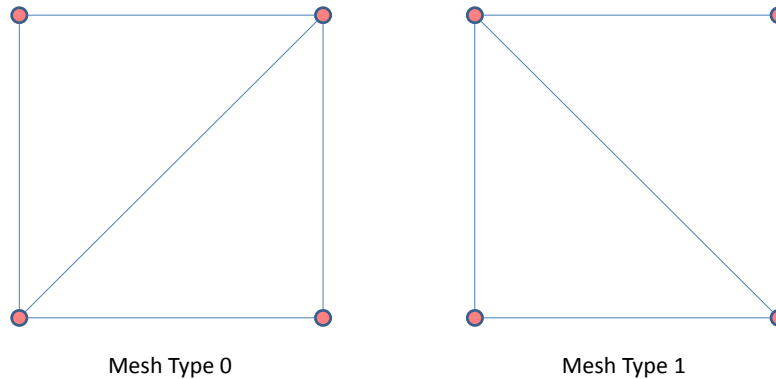
The CDB LOD structure lends itself to variable levels of terrain elevation fidelity, on a per Tile-LOD basis. The selected grid spacing is a function of the height and geographic precision that is desired. Through the use of LODs, one can specify a grid spacing appropriate to the required terrain fidelity requirements. For instance, the accurate depiction of a runway profile (say down to 1 ft height precision) would typically require a relatively fine pitch terrain elevation LOD even if the area is nominally flat. Similarly, the accurate representation of sharp altitude discontinuities (e.g., cliffs) also requires increasingly finer elevation grids to capture the cliff profile correctly.

Negative elevation values do not imply that the elevation point is submerged; rather, a negative value merely indicates that its altitude is below the WGS-84 reference ellipsoid.

The CDB Specification defines a number of subordinate elevation components that are used in combination with the primary component of the Elevation Dataset.

### 5.6.1.1 Terrain Mesh Types

The CDB Specification defines two mesh types to connect each grid post to its neighbors. The purpose of the mesh type is to minimize the error in the representation of the Terrain Profile built from the components of the Elevation dataset. Figure 5-7 below illustrates the supported CDB Mesh Types.



**Figure 5-7: CDB Mesh Types**

Mesh type 0 connects the southwest grid post to its northeast neighbor while mesh type 1 does the same for the northwest and southeast posts.

#### **5.6.1.1.1 Data Type**

The mesh type is represented by an unsigned integer of a size that is large enough to accommodate the range of mesh types defined by the Specification. As of this version of the Specification, there are only two values defined; as such, an 8-bit unsigned integer is sufficient and appropriate to store the mesh type.

#### **5.6.1.1.2 Default Value**

By default, when the mesh type is not specified or not available, a value of zero is assumed.

#### **5.6.1.2 List of all Elevation Dataset Components**

The Elevation Dataset is comprised of several components listed here and detailed in the subsequent sections.



**Table 5-11: Elevation Dataset Components**

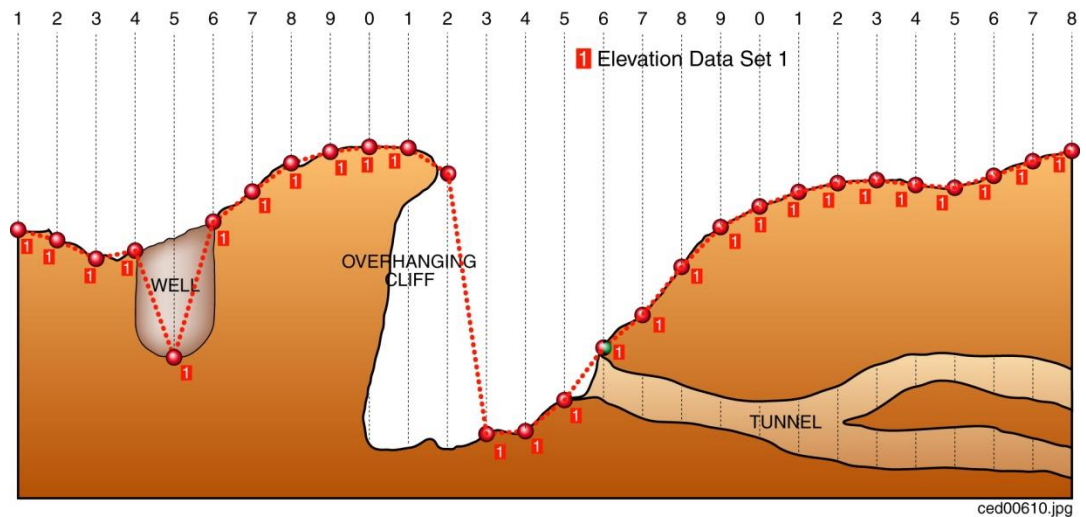
CS1	CS2	File Extension	Component Name	Component Description
<b>Dataset 001, Elevation</b>				
001	001	*.tif	Primary Terrain Elevation	A grid of data representing the Elevation at the surface of the Earth. Stored as a 1 or 2-channel TIFF image. When present, the second channel provides the mesh type of each grid element.
001	002	*.tif	Primary Terrain Elevation Control	Deprecated
001	003	*.tif	Primary Alternate Terrain Elevation	A grid of data representing the Elevation of the surface of the Earth at specified Latitude and Longitude offsets inside each grid element. Stored as a 4-channel TIFF image.
002..099	001	*.tif	Subordinate Terrain Elevation	Deprecated
002..099	002	*.tif	Subordinate Terrain Elevation Control	Deprecated
100	001	*.tif	Subordinate Bathymetry	A grid of data representing the Depth of water with respect to the selected Terrain Elevation component. Store as a 1 or 2-channel TIFF image. When present, the second channel provides the mesh type of each grid element.
100	002	*.tif	Subordinate Alternate Bathymetry	A grid of data representing the Depth of water at specified Latitude and Longitude offsets inside each grid element with respect to the selected Terrain Elevation component. Stored as a 4-channel TIFF image.
101	001	*.tif	Subordinate Tide Elevation	A grid of data representing the average height variation of water with respect to the Primary Terrain Elevation Component.
<b>Dataset 002, MinMaxElevation</b>				
001	001	*.tif	Minimum Elevation	Minimum height (on a per tile LOD basis) of the Primary Terrain Elevation Dataset Component (excluding all cultural features).
001	002	*.tif	Maximum Elevation	Maximum height (on a per tile LOD basis) of the Primary Terrain Elevation Dataset Component (excluding all cultural features).



CS1	CS2	File Extension	Component Name	Component Description
Dataset 003, MaxCulture				
001	001	*.tif	Primary Maximum Culture Elevation	Maximum height (on a per tile LOD basis) of the bounding boxes of all cultural features held in the vector tiled datasets within the geographic footprint of the area represented by the sample value.

### 5.6.1.3 Primary Terrain Elevation Component

The Primary Terrain Elevation component of the Elevation dataset represents the surface of the Earth, i.e., the emerged part of the Earth's crust, the surface of persistent bodies of water (e.g., ocean, lakes, rivers), and the permanent ice-covered parts of the Earth. However, the Primary Terrain Elevation values exclude the heights of natural vegetation and man-made cultural features.



**Figure 5-8: Primary Terrain Elevation Component**

By definition, the Primary Terrain Elevation component represents a single elevation value at each grid element of the dataset. As a result, each value of the Primary Terrain Elevation component corresponds to the elevation of the highest Earth surface at the specified latitude and longitude coordinate. Consider the example illustrated in Figure 5-8: Primary Terrain Elevation Component. The diagram illustrates a region of Earth with a well, an overhanging cliff, and a network of tunnels. Using solely the Primary Terrain Elevation component, the resulting terrain representation corresponds to the continuous terrain profile represented by the red dotted line; as a result, the underside of the overhanging cliff, the tunnels, and the vertical walls of the well are not represented.

To represent terrain walls, overhanging cliffs, wells, tunnels and mineshafts, modelers are required to supplement the Primary Terrain Elevation component with terrain-conformed 3D models as illustrated in Figure 5-9: Modeling of Wells, Overhanging



Cliffs and Tunnels. Embedded within such 3D models are special cutout zones which represent the clipping geometry that is used to cut out the terrain skin.

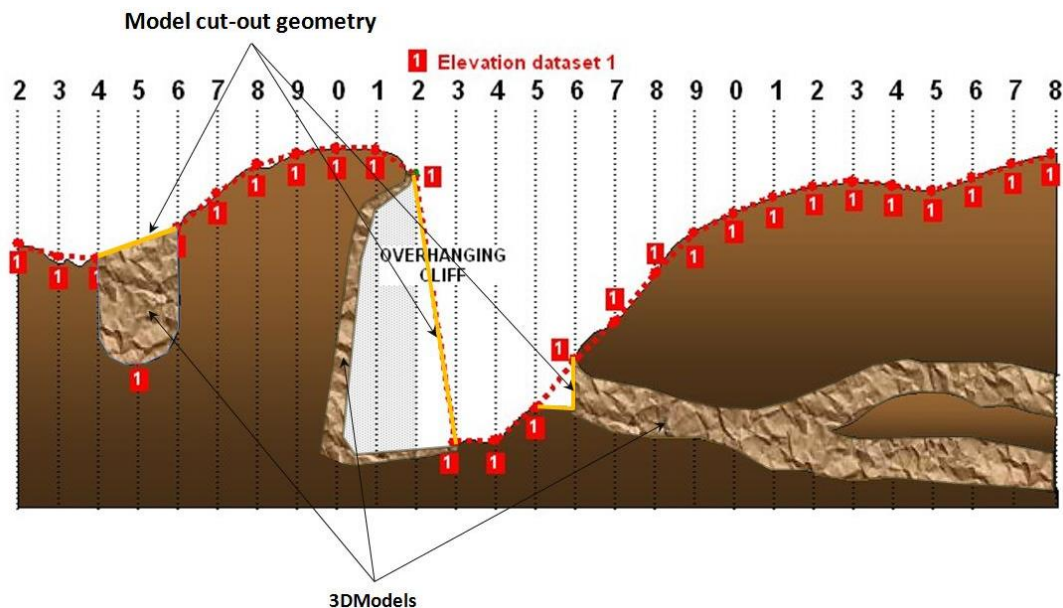


Figure 5-9: Modeling of Wells, Overhanging Cliffs and Tunnels

Model cutouts are explained in section 6.5.6.3, Model Cutout Zones and model conforming modes are described in section 6.7, Model Conforming.

#### 5.6.1.3.1 Data Type

The Primary Terrain Elevation component of the Elevation dataset is represented as a 1 or 2-channel TIFF image. The first channel contains the **Elevation** of the grid post; the optional second channel indicates the **Mesh Type** used to connect the four grid posts that are adjacent to the grid element. The elevation is represented by a floating-point or signed integer value expressed in meters and relative to the WGS-84 reference ellipsoid. Integer values for tiles at LOD larger than 0 are scaled according to the following formula:

$$Elevation = IntValue \times 2^{-LOD}$$

Integer values can make use of TIFF's 8-bit, 16-bit, or 32-bit representation.

The **Mesh Type** is stored as an unsigned 8-bit integer.

#### 5.6.1.3.2 Default Read Value

Simulator client-devices should assume an **Elevation** value of **Default\_Elevation-1** if the data values of the Primary Terrain Elevation component are not available (files

associated with the Primary Terrain Elevation component for the area covered by a tile, at a given LOD or coarser, are either missing or cannot be accessed). The default value is stored in \CDB\Metadata\Defaults.xml. In absence of a default value, the CDB Specification states that client-devices use a value of zero.

If the TIFF file has a single channel image, client devices assume a *Mesh Type* of zero.

#### 5.6.1.3.3 Default Write Value

The files associated with the Primary Terrain Elevation component for area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted. If the tool generating the Primary Terrain Elevation does not support the optional Mesh Type, the optional second channel of the file need not be created; in which case the TIFF file becomes a single channel image.

#### 5.6.1.4 Primary Alternate Terrain Elevation Component

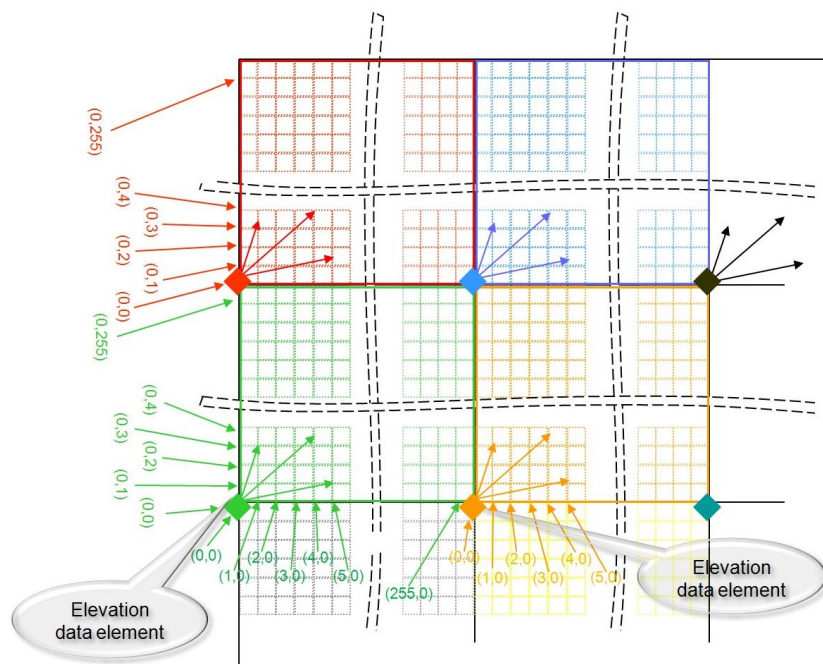
The accurate delineation of man-made elevation features such as roads, railroads, runways, and natural elevation features such as ridgelines, coastlines requires very high levels-of-detail for the Primary Terrain Elevation component. Such cases typically require an elevation grid pitch of approximately  $\frac{1}{2}$  m or better (the equivalent of 8 million triangles per square kilometer) resulting in unnecessary large storage and runtime processing. The Primary Alternate Terrain Elevation component offers an effective solution to handle these use-cases.

The Primary Alternate Terrain Elevation component provides the means to accurately delineate terrain features without having to revert to very fine LODs of the Primary Terrain Elevation component. To do this, the Primary Alternate Terrain Elevation component encodes information that re-positions each elevation sample anywhere within its assigned grid element. In other words, the “phase” of each terrain elevation sample can be specified along the latitude and longitude axes. In effect, the Primary Alternate Terrain Elevation component provides the means to locally increase the altimetric precision of the modeled representation of a terrain profile. While it would be possible for a modeler to manually control the position of individual elevation points, it is expected that the SE tools automate this process by considering elevation constraint points, lineals and areals provided by the modeler.

The constituents of the Primary Alternate Terrain Elevation are the elevation and mesh type at the specified latitude and longitude offsets inside each grid element. These four constituents are represented as 4 channels of a TIFF image.

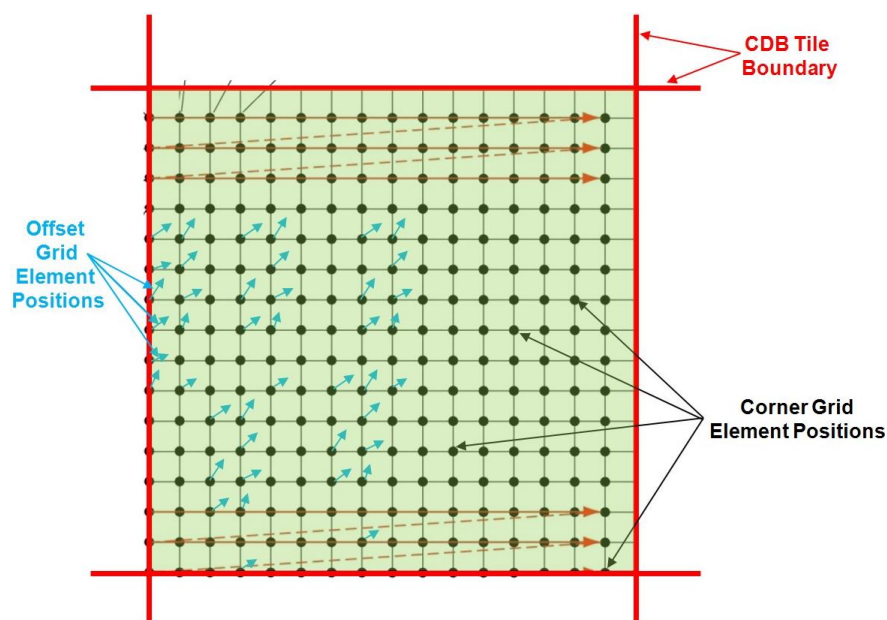
The latitude and longitude offsets are expressed as unsigned 8-bit integer values that provide position offsets expressed as 1/256th of the grid spacing for the LOD in question. Note that since the movement of each elevation point is constrained to the inside of its respective grid element, it is impossible to disrupt the (regular grid) topology of the elevation grid; furthermore, it is impossible to have elevation points

that move outside of the confines of the tile-LOD. Figure 5-10 illustrates the valid offset values inside each grid element of a tile-LOD.



**Figure 5-10: Encoding of Lat/Long Offsets**

Figure 5-11 illustrates the coverage of grid elements inside a CDB tile. It shows that a grid post is allowed to move inside the area covered by its grid element.



**Figure 5-11: Grid Element Coverage within a CDB Tile**

The **Latitude Offset** is expressed as an 8-bit unsigned integer value ranging from 0 to 255. The value is scaled so that each grid element is fragmented in 256 equal parts in

the latitude direction. Thus, the elevation point cannot be positioned on the latitude of the next grid element directly north of the current grid element.

The **Longitude Offset** is expressed as an 8-bit unsigned integer value ranging from 0 to 255. The value is scaled so that each grid element is fragmented in 256 equal parts in the longitude direction. Thus, the elevation point cannot be positioned on the longitude of the next grid element directly east of the current grid element.

#### 5.6.1.4.1 Data Type

The first channel of the TIFF image contains the **Elevation** component and is represented as a floating-point or signed integer value. Integer values for tiles at LOD larger than 0 are scaled according to the following formula:

$$Elevation = IntValue \times 2^{-LOD}$$

Integer values can make use of TIFF's 8-bit, 16-bit, or 32-bit representation.

The second channel of the TIFF image contains the **Mesh Type** and is stored as an unsigned 8-bit integer.

The third and fourth channels contain the **Latitude** and **Longitude Offset** components and are stored as unsigned 8-bit integers.

#### 5.6.1.4.2 Default Read Value

Simulator client-devices should assume an **Elevation** value of **Default\_Elevation-1** if the data values of the Primary Alternate Terrain Elevation component are not available (files associated with the Primary Alternate Terrain Elevation component for the area covered by a tile, at a given LOD or coarser, are either missing or cannot be accessed). The default value is stored in \CDB\Metadata\Defaults.xml. In absence of a default value, the CDB Specification states that client-devices use a value of zero.

The default **Mesh Type**, **Latitude** and **Longitude Offsets** are zero.

#### 5.6.1.4.3 Default Write Value

The files associated with the Primary Alternate Terrain Elevation component for an area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.

#### 5.6.1.5 Terrain Constraints

There are many instances where modelers may wish to take advantage of the availability of position and altitude of cultural features in order to locally control the terrain elevation data at a point, along a specified contour line or within a given area.

This operation is usually performed off-line by the modeler and requires that the Elevation dataset be edited and re-generated offline.

**Table 5-12: Partial List of Hypsography FACCs (for Offline Terrain Constraining)**

FACC					FACC-FSC Label	FACC Concept Definition
Level 1	Level 2	Level 3	FSC	FACC-FSC		
C					Hypsography	
	A				Relief	
		010	000	CA010-000	Contour_Line	A line connecting points having the same vertical datum value.
		020	000	CA020-000	Ridge_Line	A line representation of a ridge top.
		025	000	CA025-000	Valley_Line	A line representation of the lowest part of a valley.
		026	000	CA026-000	Breakline	Line representing the demarcation of a sudden and significant change in the gradient of the terrain relief.
		030	000	CA030-000	Spot_Elevation	A designated location with an elevation value relative to a vertical datum.
		035	000	CA035-000	Water_Elevation	A location with a generalized elevation value relative to a vertical datum associated with an inland, usually confined, water body.
		040	000	CA040-000	Contour_Polygon	An arbitrary area outline created to establish elevation as polygons.
		050	000	CA050-000	Surface_Triangle	A set of three spot elevations defining a terrain region of constant slope and aspect.

The Data Dictionary of CDB Specification makes provision for the representation of many hypsography features within the Geopolitical Dataset (see Table 5-12: Partial List of Hypsography FACCs (for Offline Terrain Constraining)). By virtue of their semantics, these features have no associated modeled representation. The modeler can use these hypsography features to control the generation of the Terrain Elevation grid during the off-line CDB compilation process. This terrain constraining operation can be performed by the SE tools as the CDB is “assembled and compiled”. Note that runtime client-devices do not constrain the Terrain Elevation Dataset to hypsography features. When terrain constraining is performed off-line, hypsography features must have AHGT set to True, thereby instructing the SE Tools to constrain the terrain elevation using the supplied (lat-long-elev) coordinates. The Shapefile feature must be a PointZ, a MultiPointZ, a PolyLineZ, a PolygonZ or a MultiPatch.

While these hypsography features can be used by the off-line SE tools to control the terrain skinning process, these features can be instead converted into Constraint Features, thereby deferring the terrain constraining process to runtime client-devices.



Constraint Features are features that instruct client-devices to runtime-constrain the terrain Elevation Dataset to a set of prescribed elevation values. This provides modelers the ability to accurately control terrain elevation profiles even if the Terrain Elevation Dataset is of modest resolution and is regularly-gridded; furthermore, the original Elevation Dataset remains unaffected. In effect, the Constraint Features provides a storage-efficient means of capturing terrain contours without having to re-generate / reskin the terrain to a higher-resolution.

Note that this operation is performed on Elevation Datasets that are regularly-gridded or irregularly-gridded. This capability is particularly effective when modelers wish to accurately control terrain elevation profiles but only have regularly-gridded source elevation data of modest resolution at their disposal. Each of these features is associated with vertices that define elevation at the supplied lat-long coordinate(s). This approach provides a level-of-control similar to that of Terrain Irregular Networks (TINs).

The following Constraint Features are used for Online Terrain Constraining:

- **Elevation Constraint Point (CA099-000):** In the case of PointZ and MultiPointZ features, the coordinates are used by the client-device to control the terrain elevation at the specified (lat-long). The Feature's AHGT attribute must be set to TRUE. Note that features implemented as Shapefile Point, PointM, MultiPoint, MultiPointM are ignored.
- **Elevation Constraint Line (CA099-001):** In the case of PolyLineZ features, the coordinates are used by the client-device to control the terrain elevation at the specified (lat-long). The Feature's AHGT attribute must be set to TRUE. Note that features implemented as Shapefile PolyLine and PolyLineM are ignored.
- **Elevation Constraint Area (CA099-002):** In the case of PolygonZ and MultiPatch features, the coordinates are used by the client-device to control the terrain elevation at the specified (lat-long). The Feature's AHGT attribute must be set to TRUE. Note that features implemented as Shapefile Polygon and PolygonM are ignored.

**Table 5-13: List of Hypsography FACCs (for Online Terrain Constraining)**

FACC					FACC-FSC Label	FACC Concept Definition
Level 1	Level 2	Level 3	FSC	FACC-FSC		
		099	000	CA099-000	Terrain_Constraint_Point	A point used to constrain the terrain elevation
		099	001	CA099-001	Terrain_Constraint_Line	A line consisting of multiple segments whose vertices and edges are used to constrain the terrain elevation
		099	002	CA099-002	Terrain_Constraint_Area	A multi-triangle convex or concave area whose vertices and edges are used to constrain the terrain elevation

An example of a point-feature is illustrated in Figure 5-12 . This picture shows a storage tank located atop a hill. Given the high terrain relief in this area, the modeler is concerned that the terrain may slope significantly in the immediate vicinity of the storage tank, particularly at coarser LODs of the uniform-sampled terrain elevation grid. As a result, he defines a PointZ Constraint Point feature that coincides with the position of the storage tank. AHGT set to True so that the client-device will constrain the Terrain Elevation dataset to the supplied value.

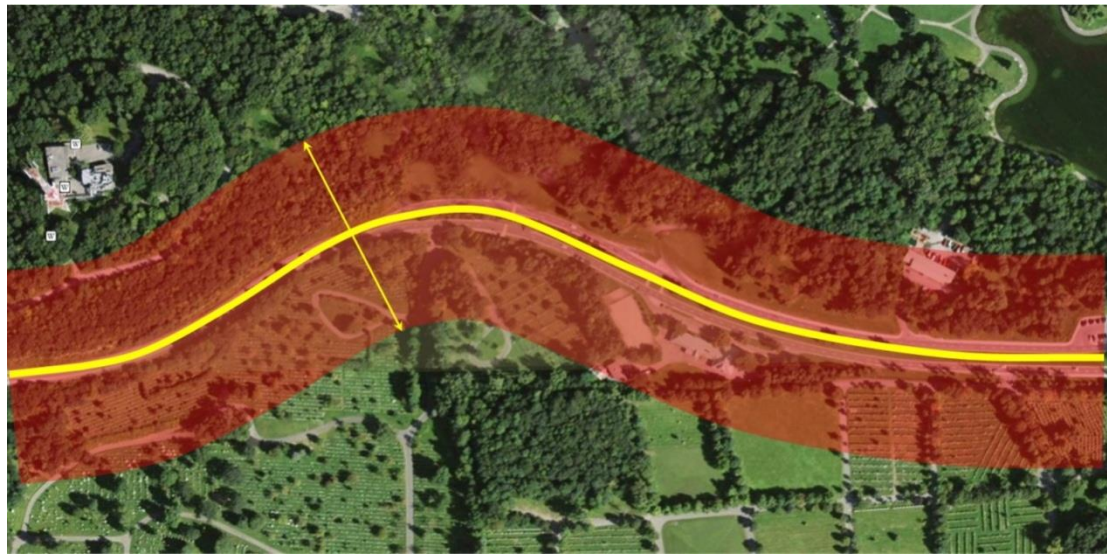


**Figure 5-12: Storage Tank Point-Feature**

A second example of this principle illustrated in Figure 5-13, this time applied to a road lineal-feature. This picture shows a divided highway running alongside a mountainous area. Given the high terrain relief in this area, the modeler is concerned that the terrain may slope significantly in the immediate vicinity of the road, particularly at the coarser LODs of the uniform-sampled terrain elevation grid. As a result, he defines a PolyLineZ Constraint Lineal feature that coincides with the centerline of the road; AHGT set to True so that the client-device will constrain the Terrain Elevation dataset to the supplied coordinates of the lineal feature.

The CDB Specification has well over 50 FACCs whose semantics are related to abstract elevation-related features (such as CA010 Contour line; CA020 Ridge line; CA025 Valley line; CA026 Breakline ...) With the exception of VG018, all of them have semantics that imply a single elevation value. The Feature “Variable Displacement Line”, FACC VG018, is an exception; it allows for a (relative) elevation value for each of the vertices of the “Variable Displacement Line”.





**Figure 5-13: Road Lineal Feature**

In the case where features overlap one other, client-devices are required to use the Layer Priority Number (LPN) attribute; this attribute is mandatory for geographically overlapping features. The LPN attribute is a number in the 0-32767; low numeric values correspond to low priority. The LPN attribute is used to control the order in which the features are applied to (e.g. rendered into) the Elevation dataset. Features are applied in succession in low-to-high priority order into the Terrain Elevation dataset.

#### **5.6.1.6 MinElevation and MaxElevation Components**

The MinElevation and MaxElevation components are part of the MinMaxElevation dataset whose purpose is to provide the CDB with the necessary data and structure to achieve a high level of determinism in computing line-of-sight intersections with the terrain. The values of each component are with respect to WGS-84 reference ellipsoid. Since both the MinElevation and the MaxElevation values are provided by this Specification, any line-of-sight algorithm can rapidly assess an intersection status of the line-of-sight vector with the terrain. An overview of the algorithm governing the line-of-sight computations can be found in Section A.13 of Appendix A.

The MinElevation and MaxElevation values follow the “center grid data element” convention of the CDB Specification.

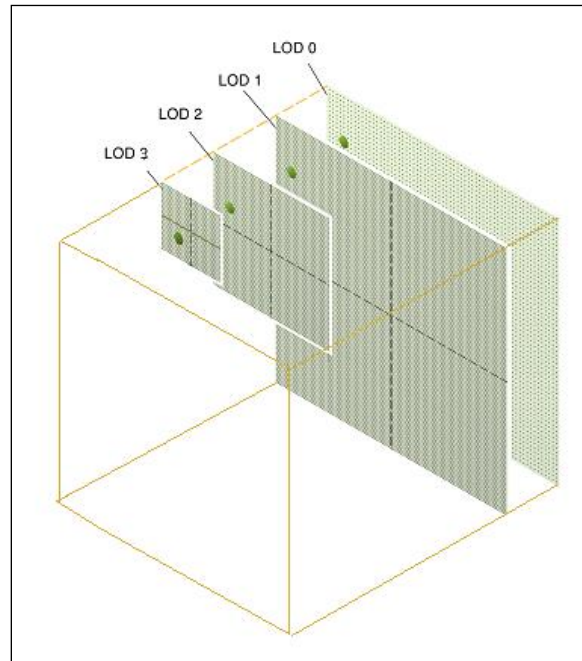
The generation of the MinMaxElevation dataset is quite simple. In essence, each center grid element in the MinElevation component represents the lowest altitude for the area represented by that grid element. Likewise, each center grid element in the MaxElevation component represents the highest altitude for the area represented by that grid element.

The MinMaxElevation dataset components are derived from the Primary Terrain Elevation and Primary Alternate Terrain Elevation components. As a result, the

MinMaxElevation dataset cannot have more LODs than the Terrain Elevation component it is based on.

#### 5.6.1.6.1 Level of Details

As can be seen in Figure 5-14: LOD Structure of Raster Datasets, the MinMaxElevation dataset LODs share the same structure as the Elevation dataset.



**Figure 5-14: LOD Structure of Raster Datasets**

The generation of each successive LOD of the MinElevation and MaxElevation components is illustrated in Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D) and again in more detail in Figure 5-16: Generation of LODs for the MinMaxElevation Dataset (2D).

The detailed algorithm for the generation of the MinMaxElevation dataset is as follows:

1. For a geocell, determine the finest available LOD of the Primary Terrain Elevation and Primary Alternate Terrain Elevation components, (call it LOD =  $n$ )
2. For each tile at LOD =  $n$ , the MinElevation (and MaxElevation) grid elements are generated by taking the corresponding minimum (and maximum) of the surrounding four “corner grid data elements” of LOD =  $n$  of the Primary Terrain Elevation component (illustrated as red dots in Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D)). If the Primary Alternate Terrain Elevation component exists at LOD =  $n$ , the value of the

Elevation must be taken into account because it provides a better estimate of the minimum or maximum elevation of the grid element. In other words, each MinElevation sample value represents the minimum for the area formed by the surrounding four “corner grid data elements” of the Primary Terrain Elevation plus the contribution of the Primary Alternate Terrain Elevation for the grid element. Likewise, each MaxElevation sample represents the maximum of the area formed by the surrounding four “corner grid data elements” of the Primary Terrain Elevation plus the contribution of the Primary Alternate Terrain Elevation for the grid element, illustrated as green dots in Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D).

Note that the generation of the rightmost (column) and topmost (row) of values of a tile requires access to the adjacent tiles of the Primary Terrain Elevation. Note however that the availability of Primary Elevation Data at  $\text{LOD} = n$  within the entire CDB geocell cannot be guaranteed since the CDB permits the generation of the Terrain Elevation Dataset at different resolutions for each geographic area as illustrated in Figure 5-18: Availability of LODs for Elevation and MinMaxElevation Datasets.

As a result, a slight adjustment to the above algorithm is needed in order to cater to the case where Elevation data is missing in adjacent tiles. There are two cases to consider:

- i. If Elevation data in the adjacent tiles (above and/or to the right) is not available at  $n \geq \text{LOD} \geq -10$ , then one or more of the 4 corner grid elements samples will be missing, hence will not be available to “participate” in the min() or max() function. In other words, the min() and max() functions must be designed to cater to a variable number of inputs depending on the availability of valid corner grid elements.
  - ii. If Elevation data in adjacent tile(s) is not available at  $\text{LOD} = n$  but is available at a coarser LOD (call it  $\text{LOD} = m$ , where  $m \geq -10$ ), then the corner grid Elevation values of the  $\text{LOD} = m$  must be propagated to finer  $\text{LOD} = n$  so that they can participate in the min() or max() functions. This principle is illustrated in Figure 5-17: Generation of LODs for the MinMaxElevation Dataset (1D) – Special Case.
3. Each grid element value of the next coarser level-of-detail ( $\text{LOD} = n-1$ ) of the MinElevation (and MaxElevation) dataset is generated by taking the minimum (and maximum) of four surrounding values of  $\text{LOD} = n$  of the MinElevation (and MaxElevation) dataset, illustrated as red dots in Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D).
  4. Repeat steps 2 and 3 for levels of detail  $\text{LOD} = n-2, n-3$ , until  $\text{LOD} -10$  is reached.
  5. Perform step 4, but this time with  $\text{LOD} = m$ , ( $m \geq -10$ ). Note that if Primary Elevation data in adjacent tile(s) is not available at  $\text{LOD} = m$  but is available

at a coarser LOD (call it  $\text{LOD} = p$ , where  $p \geq -10$ ), then the corner grid Elevation values of the  $\text{LOD} = p$  must be propagated to finer  $\text{LOD} = m$  so that they can participate in the  $\text{min}()$  or  $\text{max}()$  functions.

6. Repeat until all LODs have been processed. Note that the MaxElevation tiles at  $\text{LOD} = -10$  contain a single value which represents the highest elevation point for the entire geocell. Likewise, each of the MaxElevation tiles at  $\text{LOD} = -9$  contains four values which correspond to the highest elevation points in each of the four quadrants of the corresponding geocell.

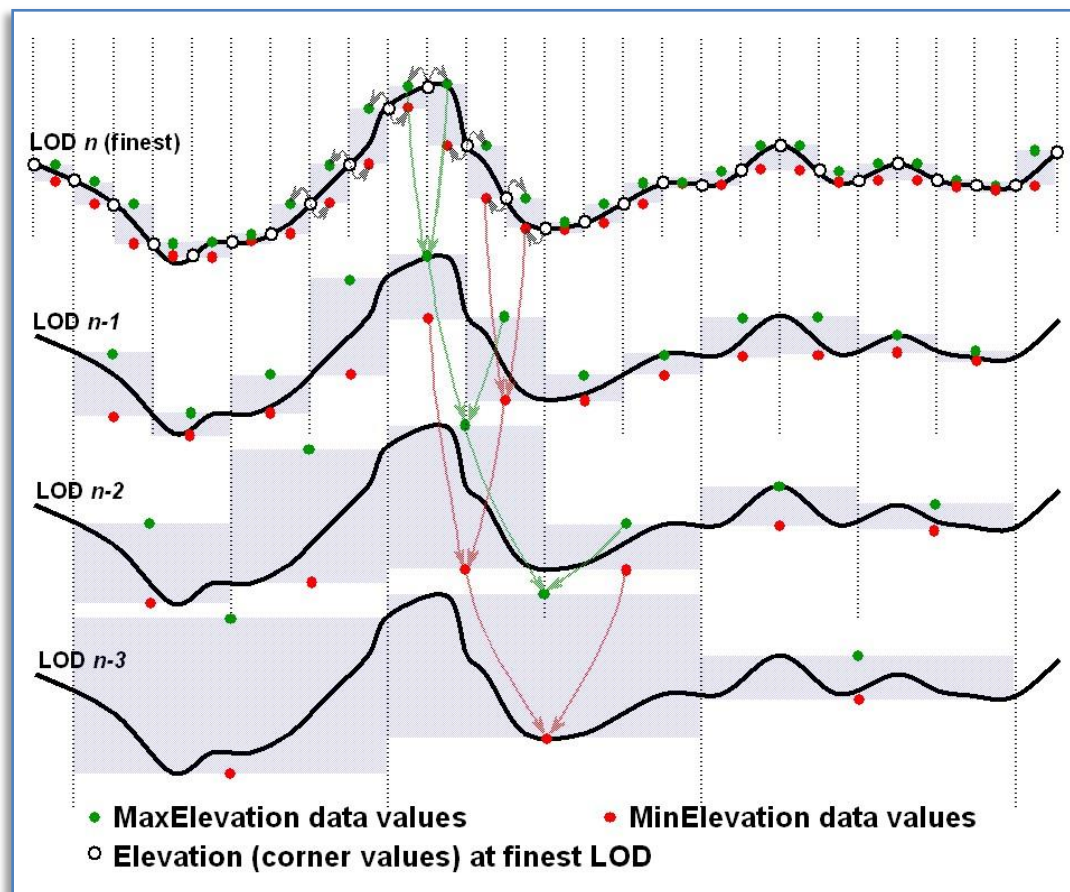


Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D)



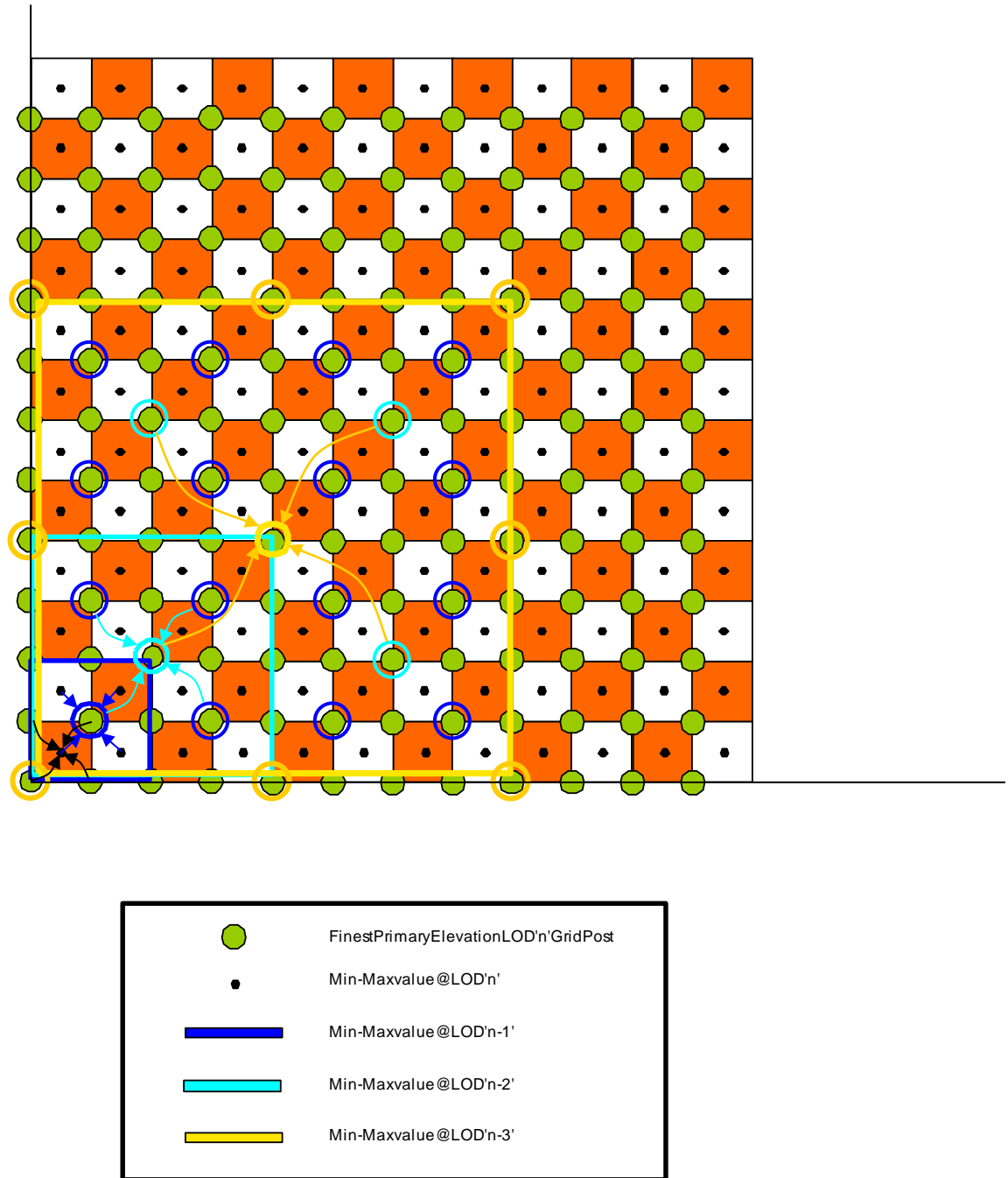


Figure 5-16: Generation of LODs for the MinMaxElevation Dataset (2D)

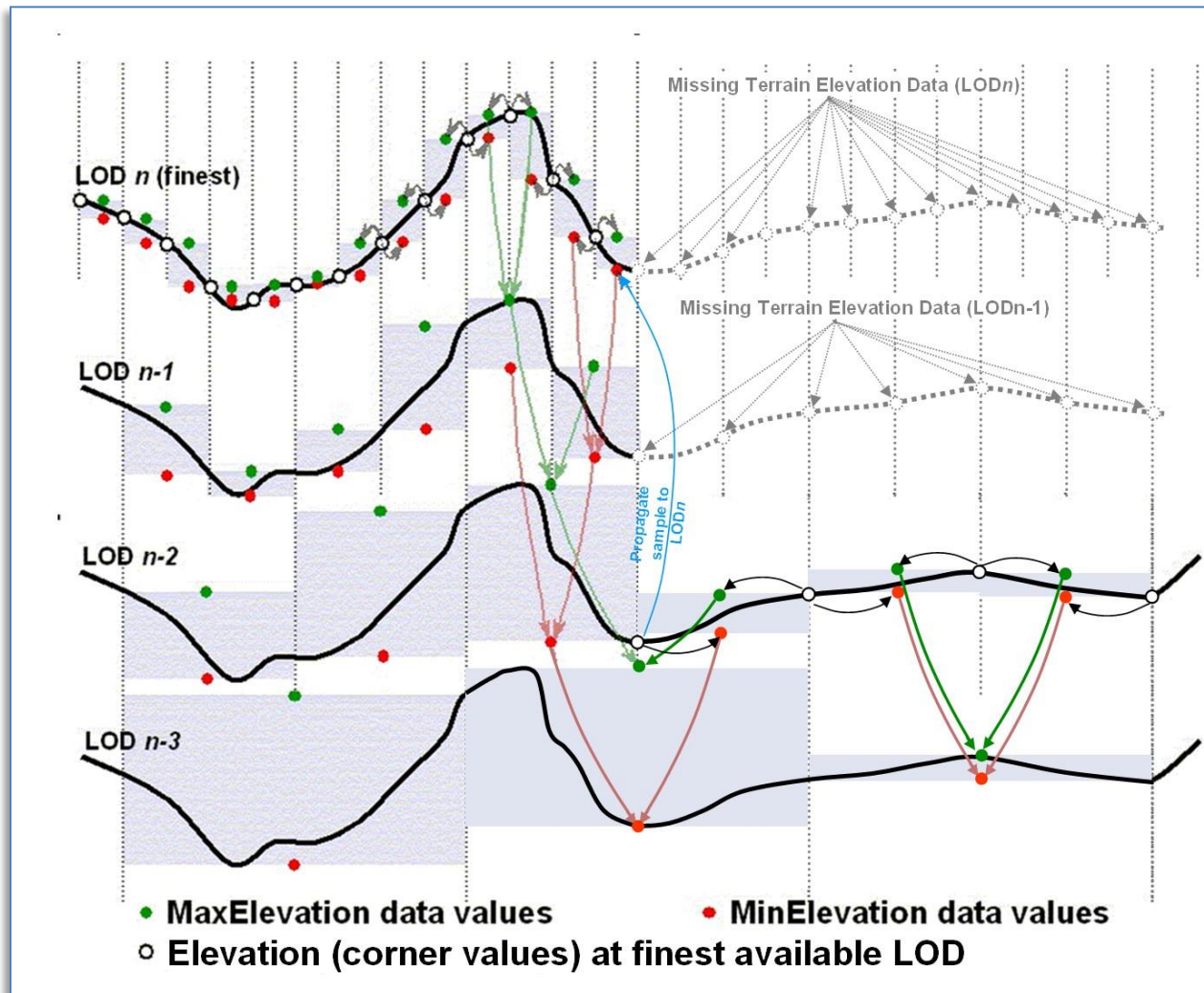
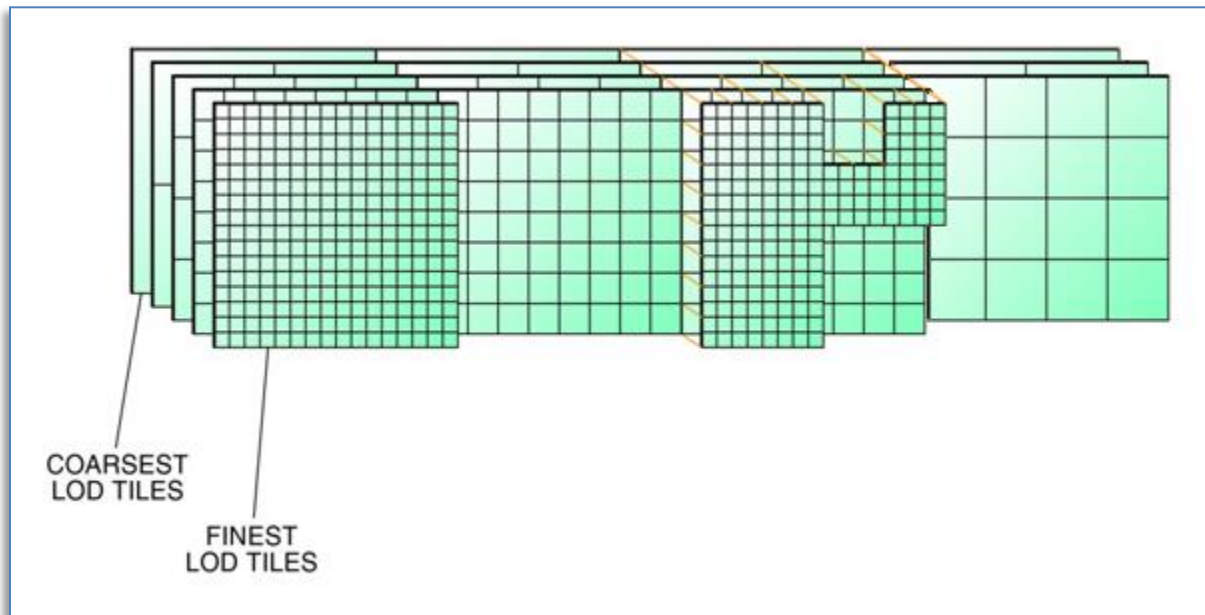


Figure 5-17: Generation of LODs for the MinMaxElevation Dataset (1D) – Special Case



**Figure 5-18: Availability of LODs for Elevation and MinMaxElevation Datasets**

The CDB Specification does not require that the entire LOD hierarchy be stored for the MinMaxElevation dataset. In fact, it is possible to omit some of the finest levels-of-detail from the hierarchy. The CDB Specification recommends that the MinElevation and MaxElevation need only be stored to  $LOD = n - 4$  and coarser (where  $n$  is the finest available LOD of the Primary Terrain Elevation component in a geocell). For example if Primary Terrain Elevation data is available for  $LOD = 15$ , then the MinMaxElevation hierarchy need only be provided for  $LOD = -10$  to  $LOD = 11$ . Note, that  $LOD = -10$  to  $LOD = 0$  are always required subject to the availability of Primary Terrain Elevation data (these guidelines are explained in more detail in section 5.6.1.6.4, Default Write Value).

Note that the presence of the MinMaxElevation dataset has a negligible effect on the size of the CDB. In fact, the dataset adds only 1% of additional storage over and above that required by the Primary Terrain Elevation component. This is a small price to pay in order to provide the means to significantly speed-up line-of-sight computations in applications requiring the utmost in determinism and real-time.

#### 5.6.1.6.2 Data Type

The MinElevation and MaxElevation components are represented as floating-point or signed integer values. Integer values for tiles at LOD larger than 0 are scaled according to the following formula:

$$Elevation = IntValue \times 2^{-LOD}$$



Integer values can make use of TIFF's 8-bit, 16-bit, or 32-bit representation.

#### **5.6.1.6.3 Default Read Value**

The Line-of-Sight algorithm is described in Section A.13 of Appendix A. Note that the algorithm starts with the coarsest LOD of the MinMaxElevation dataset; the algorithm recursively executes with progressively finer level-of-detail versions of the MinMaxElevation dataset until the algorithm decides it no longer needs to access finer levels or until the algorithm no longer finds finer levels of the MinMaxElevation dataset.

If none of the LODs of the MinMaxElevation dataset are provided, then simulator client-devices should assume default MinElevation and MaxElevation values. The default values for these datasets can be found in \CDB\Metadata\Defaults.xml and can be provided to the client-devices on demand. Handling of defaults falls under the following two cases:

1. CASE I: In the case where the tile-LOD for the MinElevation and the Primary Terrain Elevation components are both missing, the CDB Specification recommends a default setting of Default\_MinElevation\_CaseI = Default\_Elevation-1. Similarly, where a tile-LOD for MaxElevation and the Primary Terrain Elevation components are both missing, the CDB Specification recommends a default setting of Default\_MaxElevation\_CaseI = Default\_Elevation-1.
2. CASE II: In the case where the tile-LOD for the MinElevation is missing and the Primary Terrain Elevation is not missing, the CDB Specification recommends a default setting of Default\_MinElevation\_CaseII = as supplied in Defaults.xml. In the event where this default value is not supplied, the CDB Specification recommends that client-devices use a default value of -400 m (corresponding to the shore of the Dead Sea) for MinElevation.
3. Similarly, when MaxElevation is missing and the Primary Terrain Elevation is not missing, the CDB Specification recommends a default setting of Default\_MaxElevation\_CaseII = as supplied in Defaults.xml. In the event this default value is not supplied, the CDB Specification recommends that client-devices use a default value of 8846 m (corresponding to the peak of Mount Everest) for MaxElevation.

#### **5.6.1.6.4 Default Write Value**

The files associated with the MinElevation and MaxElevation components for the area covered by a tile at a given LOD should not be created if the Primary Terrain Elevation data is not available.

The CDB Specification recommends that the MinElevation and MaxElevation components be generated in accordance to the following guidelines:

1. If the finest LOD of the Primary Terrain Elevation component is available at  $\text{LOD} \geq 4$ , then all LODs ranging from  $-10 \leq \text{LOD} \leq (n - 4)$  of the MinElevation and MaxElevation components should be generated (where  $n$  is the finest available LOD of the Primary Terrain Elevation component). The technique illustrated in Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D) should be used to populate the LOD hierarchy. Gaps (i.e., missing levels) in the MIP-MAP hierarchy are not permitted. It is not permitted to generate MinElevation and MaxElevation components tiles that are partially populated with data.
2. If the finest available LOD of the Primary Terrain Elevation component is available at  $\text{LOD} \leq 3$ , then all LODs ranging from  $-10 \leq \text{LOD} \leq 0$  of the MinElevation and MaxElevation components should be generated. The technique illustrated in Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D) should be used to populate the LOD hierarchy. Gaps (i.e., missing levels) in the MIP-MAP hierarchy are not permitted. It is not permitted to generate MinElevation and MaxElevation components tiles that are partially populated with data.

In the event where parts of a MinElevation tile cannot be determined due to missing primary elevation tiles, the CDB Specification recommends to use a default value of Default\_MinElevation\_CaseIII, -400 m (corresponding to the shore of the Dead Sea) for MinElevation. Similarly, in the event where parts of a MaxElevation tile cannot be determined due to missing primary elevation tiles, the CDB Specification recommends to use a default value of Default\_MaxElevation\_CaseIII, 8846 m (corresponding to the peak of Mount Everest) for MaxElevation. Figure 5-19: Missing MinMaxElevation Datasets shows this case:

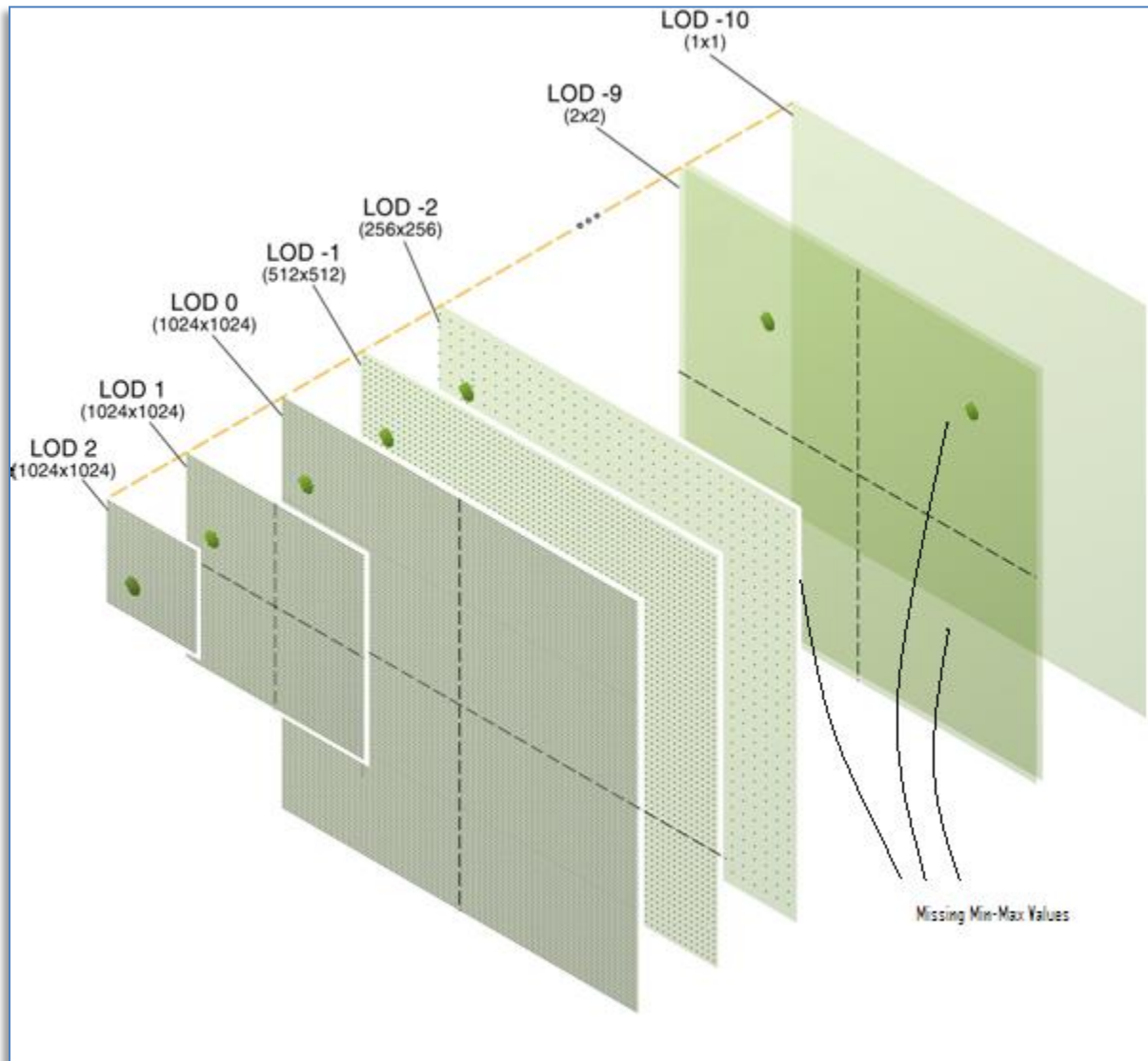


Figure 5-19: Missing MinMaxElevation Datasets

### 5.6.1.7 MaxCulture Component

The purpose of the MaxCulture component is to provide the necessary data and structure for an optimal level of determinism in the computation of line-of-sight, path finding and obstacle avoidance algorithms with the cultural features of the CDB. The values of this component are based on the heights of culture features with respect to the corresponding LOD of the culture, be it its bounding sphere, its bounding box or its modeled representation (if supplied). In this context, the cultural features of the CDB are those represented by all Tiled Vector Datasets (see Section 5.7) excluding those related to NAV.

Since MinElevation, MaxElevation and MaxCulture components are provided by this Specification, any line-of-sight algorithm can rapidly assess an intersection status of

the line-of-sight vector with the terrain and/or with the cultural features of the CDB. Furthermore, since the MaxCulture component follows the same conventions as the MinElevation and MaxElevation components, it is easy for the LOS algorithm to combine the values to determine the highest/lowest point (with or without cultural features) in a given geographic area. The culture-variant of the LOS algorithm is virtually identical to the terrain-only case. Before undertaking its computations, the LOS algorithm must add the values of MaxCulture to the MaxElevation values, once adjusted for LOD, and then perform the first LOS determination based on this. If an intersection is detected with MaxCulture, the final determination of intersection is conducted at first with the bounding box of the cultural feature, then with the actual geometry of the cultural feature (if available).

Note that the geographic areas where MaxCulture is zero can be used to quickly identify the absence of any obstacles that can potentially affect the route of an entity.

The MaxCulture component also follows the “center grid data element” convention of the CDB Specification. In the case where a cultural feature has no modeled representation, the MaxCulture component must be generated from the feature’s bounding volume that overlaps each MaxCulture grid data element. If the feature has an associated modeled representation, the grid data of the MaxCulture component must be generated from the model geometry.

#### **5.6.1.7.1 Level of Details**

The coarser LODs of the MaxCulture component are iteratively derived from the finest generated LOD.

Since the MaxCulture component is intended to be used in conjunction with the MaxElevation component, it is recommended that the number of LODs for the MaxCulture component be equal or greater than the MaxElevation component.

#### **5.6.1.7.2 Data Type**

The MaxCulture component is represented as floating-point or signed integer values. Integer values for tiles at LOD larger than 0 are scaled according to the following formula:

$$Elevation = IntValue \times 2^{-LOD}$$

Integer values can make use of TIFF’s 8-bit, 16-bit, or 32-bit representation.

#### **5.6.1.7.3 Default Read Value**

If none of the LODs of the MaxCulture dataset are provided, then simulator client-devices should assume default MaxCulture values. The default values for these

datasets can be found in \CDB\Metadata\Defaults.xml and can be provided to the client-devices on demand. Handling of defaults fall under the following two cases:

1. CASE I: In the case where the MaxCulture component is missing, but there is at least one vector dataset; client-devices should assume a default MaxCulture value of Default\_MaxCulture\_CaseI. In the event this default value is not supplied, the CDB Specification recommends that client-devices use a value of 600 m (corresponding to the tip of World's tallest tower plus a margin of 47 m).
2. CASE II: In the case where the MaxCulture component is missing, but there is not a single vector dataset; client-devices should assume a default MaxCulture value of Default\_MaxCulture\_CaseII. In the event this default value is not supplied, the CDB Specification recommends that client-devices use a value of 0 m.

#### **5.6.1.7.4 Default Write Value**

The files associated with the MaxCulture components for the area covered by a tile at a given LOD should not be created if the Primary Terrain Elevation data is not available.

The CDB Specification strongly recommends that the MaxCulture dataset be generated in accordance to the following guidelines:

1. If the finest LOD of any vector tiled datasets is available at  $\text{LOD} \geq 6$ , then all LODs ranging from  $-10 \leq \text{LOD} \leq (n - 6)$  of the MaxCulture dataset should be generated (where  $n$  is the finest available LOD of any vector tiled datasets). The technique illustrated in Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D) should be used to populate the LOD hierarchy. Gaps (i.e., missing levels) in the MIP-MAP hierarchy are not permitted. It is not permitted to generate MaxCulture dataset tiles that are partially populated with data.
2. If the finest LOD of any vector tiled datasets is available at  $\text{LOD} \leq 5$ , then all LODs ranging from  $-10 \leq \text{LOD} \leq 0$  of the MaxCulture dataset should be generated (where  $n$  is the finest available LOD of any vector tiled datasets). The technique illustrated in Figure 5-15: Generation of LODs for the MinMaxElevation Dataset (1D) should be used to populate the LOD hierarchy. Gaps (i.e., missing levels) in the MIP-MAP hierarchy are not permitted. It is not permitted to generate MaxCulture dataset tiles that are partially populated with data.

#### **5.6.1.8 Subordinate Bathymetry Component**

The Subordinate Bathymetry component consists of a grid of data values that represent the depth of water (be it of fresh water bodies or of the ocean) with respect to the corresponding data values of the Terrain Elevation (be it the Primary Terrain

Elevation or Primary Alternate Terrain Elevation components). The Subordinate Bathymetry component follows the corner grid element conventions.

The generation of Bathymetry values is best explained by the illustration found in Figure 5-20: Primary Terrain Elevation and Subordinate Bathymetry Components. In areas where Primary Terrain Elevation values correspond to the surface of a body of water, each Bathymetry value represents the height difference between the corresponding Primary Terrain Elevation value (the reference) and the Earth's Crust. In all other areas, the Bathymetry values represent the height difference between the nearby water body and the Earth's Crust. Appendix A.8 provides the mandated behavior of client-devices when reading a LOD of a primary component and combining it with another LOD of a subordinate component such as the Bathymetry.

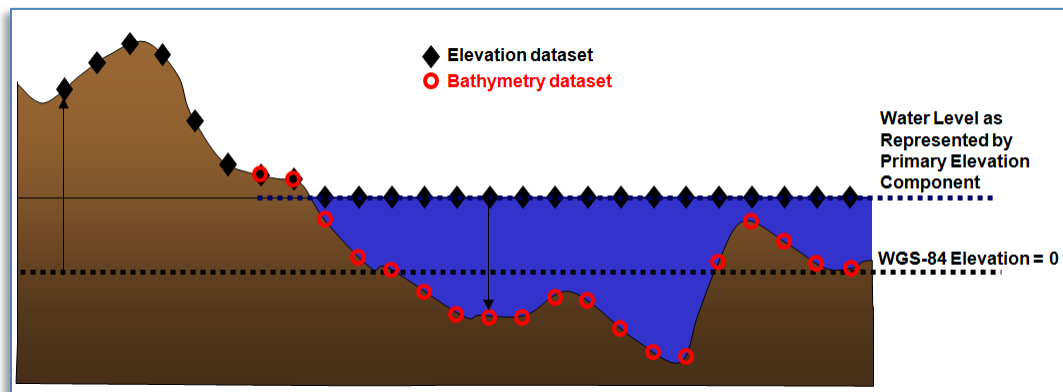


Figure 5-20: Primary Terrain Elevation and Subordinate Bathymetry Components

Positive (depth) values of Bathymetry indicate that the corresponding grid element is submerged, i.e., the Earth's Crust is below the elevation values in the Primary Terrain Elevation component. Zero values correspond to the shoreline of the water body. Negative values of Bathymetry indicate that the grid element is above water. The use of negative values of Bathymetry in the vicinity of shorelines provides a better means of interpolating Bathymetry along shorelines; such interpolation improve the precision of shoreline contours that can be derived from the isoline  $B = 0$ .

In areas that are submerged, the Primary Terrain Elevation component represents the surface of the water, not the elevation of the Earth's Crust. The height of any point of the Earth's Crust with respect to the WGS-84 reference ellipsoid can be determined using Equation (eq. 5-3).

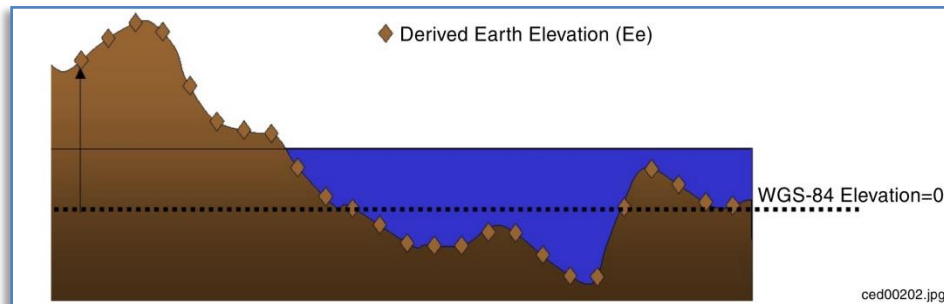
$$E_e = E - \max(0, B) \quad (\text{eq. 5-3})$$

Where  $E$  = Terrain Elevation component

$B$  = Bathymetry component

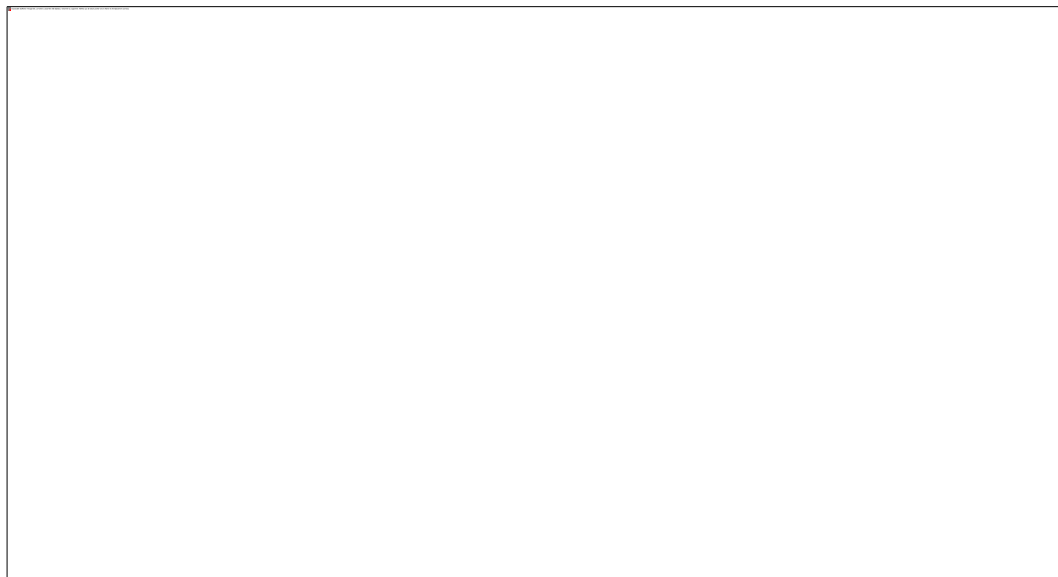
$E_e$  = Earth's Crust Elevation

The resulting profile of the Earth's Crust is shown in Figure 5-21: Derived Earth Elevation Values.



**Figure 5-21: Derived Earth Elevation Values**

The Bathymetry component needs to be provided only in areas on the Earth's surface where water is present. Below is another example of the relations between the Primary Elevation component and the Subordinate Bathymetry component.



**Figure 5-22: Example of Primary Terrain Elevation and Bathymetry Components**

#### 5.6.1.8.1 Data Type

The Subordinate Bathymetry component of the Elevation dataset is represented as a 1 or 2-channel TIFF image. The first channel contains the **Depth** of the grid post; the optional second channel indicates the **Type of Mesh** used to connect the four grid posts that are adjacent to the grid element. The depth is represented by a floating-point or signed integer value expressed in meters and relative to the Primary Terrain



Elevation component. Integer values for tiles at LOD larger than 0 are scaled according to the following formula:

$$\text{Depth} = \text{Intvalue} \times 2^{-\text{LOD}}$$

Integer values can make use of TIFF's 8-bit, 16-bit, or 32-bit representation.

The **Mesh Type** is stored as an unsigned 8-bit integer.

#### 5.6.1.8.2 Default Read Value

Simulator client-devices should assume a **Depth** value of **Default\_Bathymetry** if the data values are not available. The default value can be found in \CDB\Metadata\Defaults.xml. In the case where the default value cannot be found, the CDB Specification states that client-devices use a value of zero. The default **Mesh Type** is zero.

#### 5.6.1.8.3 Default Write Value

The files associated with the Subordinate Bathymetry component for area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted. If the tool generating the Subordinate Bathymetry component does not support the optional Mesh Type, the optional second channel of the file need not be created; in which case the TIFF file becomes a single channel image.

#### 5.6.1.9 Subordinate Alternate Bathymetry Component

The Subordinate Alternate Bathymetry component is similar to the Primary Alternate Terrain Elevation component; it provides a better delineation of the shoreline and bottom of water bodies such as oceans, lakes, and rivers. To do this, the Subordinate Alternate Bathymetry component encodes information that re-positions each depth samples anywhere within its assigned grid element. In other words, the “phase” of each bathymetry depth sample can be specified along the latitude and longitude axes. In effect, the Subordinate Alternate Bathymetry component provides the means to locally increase the precision of the modeled representation of the floor of water bodies. Again, it is expected that the SE tools produce the Subordinate Alternate Bathymetry component by considering constraint points, lineals and areals provided by the modeler.

The constituents of the Subordinate Alternate Bathymetry are the depth and mesh type at the specified latitude and longitude offsets inside each grid element. These four constituents are represented as 4 channels of a TIFF image.

#### 5.6.1.9.1 Data Type

The first channel of the TIFF image contains the **Depth** component and is represented as a floating-point or signed integer value. Integer values for tiles at LOD larger than 0 are scaled according to the following formula:

$$Depth = Sample \times 2^{-LOD}$$

Integer samples can make use of TIFF's 8-bit, 16-bit, or 32-bit representation.

The second channel of the TIFF image contains the **Mesh Type** and is stored as an unsigned 8-bit integer.

The third channel of the TIFF image contains the **Latitude Offset** and is stored as an 8-bit unsigned integer value ranging from 0 to 255. The value is scaled so that each grid element is fragmented in 256 equal parts in the latitude direction. Thus, the grid post cannot be positioned on the latitude of the next grid element directly north of the current grid element.

The fourth channel of the TIFF image contains the **Longitude Offset** and is stored as an 8-bit unsigned integer value ranging from 0 to 255. The value is scaled so that each grid element is fragmented in 256 equal parts in the longitude direction. Thus, the grid post cannot be positioned on the longitude of the next grid element directly east of the current grid element.

#### 5.6.1.9.2 Default Read Value

Simulator client-devices should assume a **Depth** of zero (as well as a **Latitude** and **Longitude Offsets** of zero and a **Mesh Type** of zero) when the Subordinate Alternate Bathymetry component is not available.

#### 5.6.1.9.3 Default Write Value

The files associated with the Subordinate Alternate Bathymetry component for an area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.

#### 5.6.1.10 Subordinate Tide Component

The Tide component represents the height variation of water (be it of fresh water bodies of water or of the ocean) with respect to the Primary Elevation component. The Tide component implicitly follows the corner grid element conventions. Each value in the Tide component must be matched to the available LOD elevation values of the Primary Elevation component.

The Tide component needs only to be provided in areas on the Earth's surface that are in the vicinity of water bodies. The information held in the Terrain Elevation and

Tide components permits a means for client-devices to accurately determine the shoreline profile as a function of the tide level. When provided, the Tide component permits client devices to compute the elevation (with respect to the WGS-84 mean sea-level reference) in areas permanently or potentially submerged. The Tide component need not be limited to oceans; it can also be used to specify the variation of height of any body of water (rivers, lakes, gulfs, etc.).

The Tide component also permits simulation of tides that varies with location. In order to determine the shoreline profile at a given location, the simulator client-devices must first determine the height of (say) the ocean in the immediate vicinity of that location. The sophistication of this calculation can vary greatly with simulation fidelity. A discussion of possible alternatives regarding the fidelity of simulation Tide simulation models can be found in Section A.8 of Appendix A.

With the CDB Tide component, simulator client-devices can readily determine the height of the ocean (or any water surface whose height varies) at any point and as a result can derive the geometry of the shoreline<sup>61</sup>. While a stored vector shoreline representation might provide a more straightforward means of representing the shoreline geometry for some client-devices, that representation would not lend itself to determining the variation of the shoreline geometry with varying tides. Furthermore, a vectorized representation of the shoreline geometry would essentially be a single-level of detail of the shoreline geometry; as a result, it would need to be generated at a resolution designed to match the highest LOD Terrain Elevation data. Coarser shoreline LODs would essentially be samples of the shoreline vector geometry at progressively greater spatial intervals.

The CDB Tide component represents the height variation of water surfaces anywhere on the Earth's surface. The variation need not be limited to the effect of tides<sup>62</sup>. The Tide component represents the height variation of the water surface above and below the mean water surface level.

---

<sup>61</sup> While a stored vector shoreline representation might have provided a more straightforward means of representing the shoreline geometry for some client-devices, that representation was rejected because it would not lend itself to determining the variation of the shoreline geometry with varying tides.

<sup>62</sup> For instance, it could represent the nominal seasonal variations of water level of lakes and rivers.

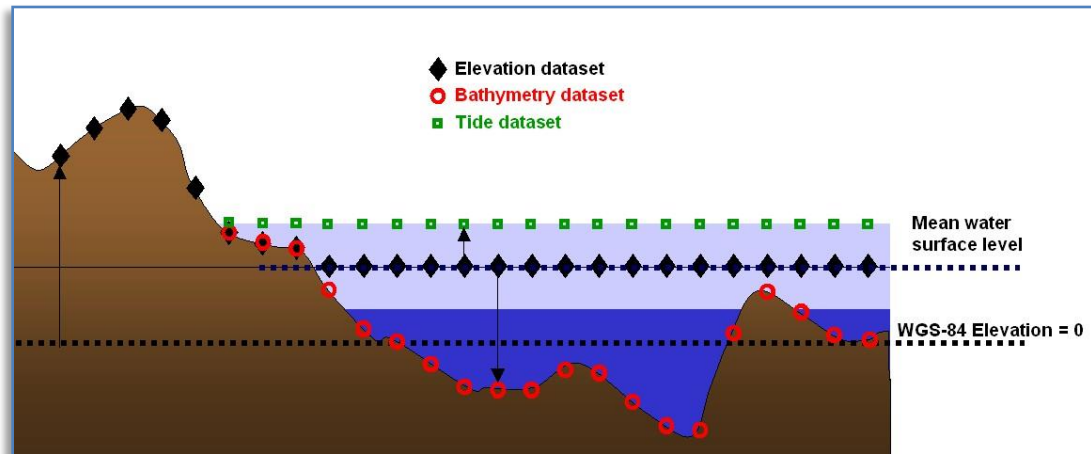


Figure 5-23: Terrain Elevation, Bathymetry and Tide Components

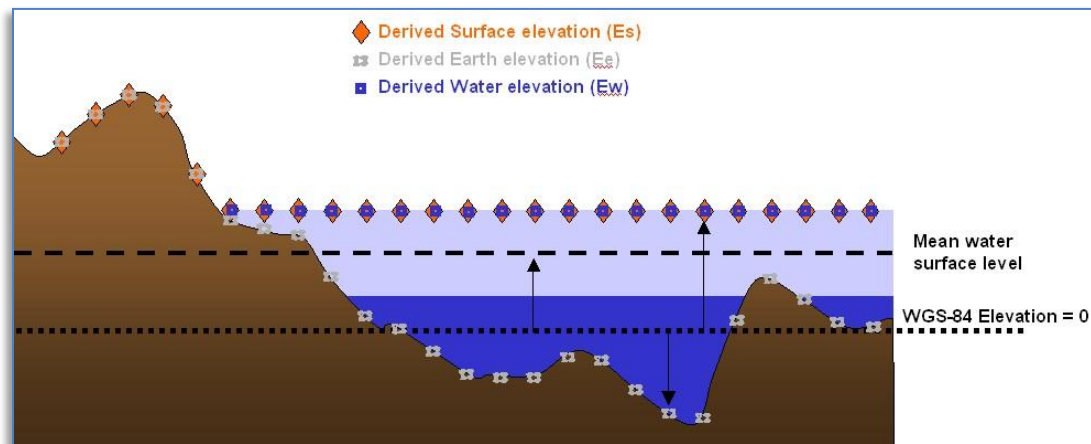


Figure 5-24: Derived Earth Elevation, Water Elevation and Surface Elevation Values

From the above components, simulation client devices can compute a) the elevation of the water  $E_w$ , b) the elevation of the earth's surface  $E_e$  (be it submerged or potentially submerged), and c) the surface elevation of the earth / water  $E_s$ . These computations can be performed in all areas where the Bathymetry and Tide components are available (e.g., areas submerged or potentially submerged). The values for  $E_w$ ,  $E_e$ , and  $E_s$  are referenced to the WGS-84 mean sea-level reference level. Equation (eq. 5-4) though (eq. 5-6) can be used to compute  $E_w$ ,  $E_e$  and  $E_s$ :

$$E_w = E + \min(0, B) + T \quad (\text{eq. 5-4})$$

$$E_e = E - \max(0, B) \quad (\text{eq. 5-5})$$

$$E_s = \max(E_e, E_w) \quad (\text{eq. 5-6})$$

Where  $E$  = Terrain Elevation component

$B$  = Bathymetry component

$T$  = Tide component

$E_w$  = Derived water elevation value

$E_e$  = Derived earth elevation value

$E_s$  = Derived surface elevation

Client devices interested in computing the height of the ownship over terrain or water can use equation (eq. 5-7).

$$HAT = O - E_s \quad (\text{eq. 5-7})$$

Where  $O$  = Ownship Altitude

Finally, client devices interested in determining the depth of water  $D$ , can use equation (eq. 5-8).

$$D = \min(0, E_w - E_e) \quad (\text{eq. 5-8})$$

**NOTE:** A computed value of  $D$  of 0 means the point is above water.

#### 5.6.1.10.1 Data Type

Tide components are represented as floating-point or signed integer values. Integer values for tiles at LOD larger than 0 are scaled according to the following formula:

$$\text{Tide} = \text{Intvalue} \times 2^{-\text{LOD}}$$

Integer values can make use of TIFF's 8-bit, 16-bit, or 32-bit representation.

#### 5.6.1.10.2 Default Read Value

Simulator client-devices should assume default Tide values if the data values are not available (files associated with the Tide component for the area covered by a tile, at a given LOD or coarser, are either missing or cannot be accessed). The default value can be provided to the client-devices on demand. The CDB Specification recommends a default tide value of 2.5m (published average magnitude of tides worldwide).

Simulator client-devices should assume a default Tide value of Default\_Tide if the data values are not available (files associated with the Tide component for the area covered by a tile, at a given LOD or coarser, are either missing or cannot be

accessed). The default value can be found in \CDB\Metadata\Defaults.xml and can be provided to the client-devices on demand. In the case where the default value cannot be found, the CDB Specification recommends that client-devices use a default tide value of 2.5m (average magnitude of tides worldwide).

#### **5.6.1.10.3 Default Write Value**

The files associated with the Tide component for area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.

### **5.6.2 Tiled Imagery Dataset**

In a CDB, the terrain imagery is depicted on a grid at regular geographic intervals. Each of the components of the Imagery Dataset corresponds to the raster imagery draped over the terrain skin derived from the Primary Terrain Elevation Dataset. The Raster Imagery Dataset implicitly follows the center grid element conventions.

The CDB Specification provides for a set of alternate terrain imagery representations corresponding to the visible spectrum terrain imagery at different periods of the year. Together, these representations are stored in a set of Visible Spectrum Terrain Imagery (VSTI) components. Each of these representations can be either monochrome or color.

In addition, the CDB Specification provides for a subordinate light map representation that can be applied to the selected VSTI component for a night-time representation of lighting patterns created by the projection of light-sources onto the terrain surface. The light-map can be either monochrome or color.

#### **5.6.2.1 Raster-Based Imagery File Storage Extension Naming**

As briefly mentioned earlier in Section 1.4.10, the CDB Specification introduces the notion of support for JPEG 2000 raster-based storage format for raster imagery files. Since the CDB Specification enforces a unique filename for each dataset file, a different file extension is required for such a dataset file format to distinguish it from TIFF for other raster based datasets, thus any raster imagery dataset shall be stored under the “.jp2” file extension.

##### **5.6.2.1.1 JPEG 2000 Metadata**

In addition to the compressed image data, the JPEG 2000 files may contain metadata to hold additional data boxes. They are the Intellectual Property box, XML box, URL box and UUID box. Among them, the XML box is perfectly suited to store formatted metadata concerning the source of this data, or the security attributes associated with the file usage. Below is the XML format description of such metadata to be supported as part of this CDB Specification. It is to be noted that the existence of this XML metadata box does not contain any information necessary for decoding the

image portion, and the correct interpretation of the XML data will not change the visual appearance of the image. This metadata is divided in two distinct elements, namely ORIGIN and SECURITY.

#### 5.6.2.1.1.1 Origin of data

XML Tag Name	Format	Description	Values
datetime	STRING	File Date & Time: This field shall contain the file's origination in the format CCYYMMDDhhmmss, where CC is the first two digits of the year (00-99), YY is the last two digits of the year (00-99), MM is the month (01-012), DD is the day (01-31), hh is the hour (00-23), mm is the minute (00-59), and ss is the second (00-59). UTC is assumed to be the time zone designator to express the time of day.	Default is 14 spaces  Date Format: CCYYMMDDhhmmss
originatingstationid	STRING	Originating Station ID: This field shall contain the identification code or name of the originating organization, system, station, or product. It shall not be filled with spaces.	This 10-character field must NOT be blank
originatorname	STRING	Originator's Name: This field shall contain a valid name for the operator who originated the file. If the field is all spaces, it shall represent that no operator is assigned responsibility for origination.	Default is 24 spaces
originatorphone	STRING	Originator's Phone Number. This field shall contain valid phone number for the operator who originated the file. If the field is all spaces, it shall represent that no phone number is available for the operator assigned responsibility for origination.	Default is 18 spaces
originatororganization	STRING	Originator's Organization. This field shall contain a valid name of the supporting organization.	Default is 80 spaces
originatoraddress	STRING	Originator's Address. This field shall contain a valid address of the supporting organization.	Default is 256 spaces
originatoremail	STRING	Originator's Electronic Mail Address. This field shall contain a valid email address of the supporting organization.	Default is 100 spaces
originatorwebsite	STRING	Originator's Web Site Address. This field shall contain a valid web site address of the supporting organization.	Default is 100 spaces
originatorremark	STRING	Originator's Remark Text. This field shall contain description text for any special remarks concerning the file.	Default is 100 spaces



**5.6.2.1.1.2 Security**

Attribute Name	Format	Description	Values
classificationlevel	BYTE	File Security Classification: This field shall contain a 1-character valid value representing the classification level of the entire file.	Valid values are: T (=Top Secret), S (=Secret), C (=Confidential), R (=Restricted), U (=Unclassified).
system	STRING	File Security Classification System: This field shall contain valid values indicating the national or multinational security system used to classify the file. If this field is all blank spaces, it shall imply that no security classification system applies to the file.	Default is 2 spaces  Country Codes per FIPS 10-4 shall be used to indicate national security systems; codes found in DIAM 65-19 shall be used to indicate multinational security systems.
codewords	STRING	File Codewords: This field shall contain a valid indicator of the security compartments associated with the file.	Default is 11 spaces
controlhandling	STRING	File Control and Handling. This field shall contain valid additional security control and/or handling instructions (caveats) associated with the file. Values include digraphs found in DIAM 65-19 and/or MIL_STD_2500B-Table A-4. The digraph may indicate single or multiple caveats. The selection of a relevant caveat(s) is application specific. If this field is all spaces, it shall imply that no additional control and handling instructions apply to the file.	Default is 2 spaces  Values include one or more of the tri/digraphs found in DIAM 65-19 and/or MIL_STD_2500B-Table A-4. Multiple entries shall be separated by a single space
releaseinstructions	STRING	File Releasing Instructions. This field shall contain a valid list of country and/or multilateral entity codes to which countries and/or multilateral entities the file is authorized for release. If this field is all spaces, it shall imply that no file release instructions apply.	Default is 20 spaces  Valid items in the list are one or more country codes as found in FIPS 10-4 and/or codes identifying multilateral entities as found in DIAM 65-19.

Attribute Name	Format	Description	Values
declassificationtype	STRING	<p>File Declassification Type. This field shall contain a valid indicator of the type of security declassification or downgrading instructions which apply to the file.</p> <p>If this field is all spaces, it shall imply that no file security declassification or downgrading instructions apply.</p>	<p>Default is 2 spaces</p> <p>Valid values are: DD (=declassify on a specific date), DE (=declassify upon occurrence of an event), GD (=downgrade to a specified level on a specific date), GE (=downgrade to a specified level upon occurrence of an event), O (=OADR), X (= exempt from automatic declassification).</p>
declassificationdate	STRING	<p>File Declassification Date. This field shall indicate the date on which a file is to be declassified if the value in File Declassification Type is DD. If this field is all spaces, it shall imply that no file declassification date applies.</p>	<p>Default is 8 spaces</p> <p>Date Format: CCYYMMDD</p>
declassificationexemption	STRING	<p>File Declassification Exemption. This field shall indicate the reason the file is exempt from automatic declassification if the value in File Declassification Type is X.</p> <p>If this field is all spaces, it shall imply that a file declassification exemption does not apply.</p>	<p>Default is 4 spaces</p> <p>Valid values are X1 through X8 and X251 through X259. X1 through X8 correspond to the declassification exemptions found in DOD 5200.1-R, paragraphs 4-202b(1) through (8) for material exempt from the 10-year rule. X251 through X259 correspond to the declassification exemptions found in DOD 5200.1-R, paragraphs 4-301a(1) through (9) for permanently valuable material exempt from the 25-year declassification system.</p>
filedowngrade	BYTE	<p>File Downgrade. This field shall indicate the classification level to which a file is to be downgraded if the values in File Declassification Type are GD or GE. If this field is all spaces, it shall imply that file security downgrading does not apply.</p>	<p>Default is 1 space</p> <p>Valid values are: S (=Secret), C (=Confidential), R (=Restricted).</p>



Attribute Name	Format	Description	Values
filedowngradedate	STRING	File Downgrade Date. This field shall indicate the date on which a file is to be downgraded if the value in File Declassification Type is GD. If this field is all spaces, it shall imply that a file security downgrading date does not apply.	Default is 8 spaces  Date Format: CCYYMMDD
classificationtext	STRING	File Classification Text. This field shall be used to provide additional information about file classification to include identification of declassification or downgrading event if the values in File Declassification Type are DE or GE. It may also be used to identify multiple classification sources and/or any other special handling rules. If this field is all spaces, it shall imply that additional information about file classification does not apply.	Default is 43 spaces  Values are user defined free text.
classificationauthoritytype	BYTE	File Classification Authority Type. This field shall indicate the type of authority used to classify the file. If this field is all spaces, it shall imply that file classification authority type does not apply.	Default is 1 single space  Valid values are: O (= original classification authority), D (= derivative from a single source), M (=derivative from multiple sources).

Attribute Name	Format	Description	Values
classificationauthority	STRING	File Classification Authority: This field shall identify the classification authority for the file dependent upon the value in File Classification Authority Type. If this field is all spaces, it shall imply that no file classification authority applies.	Default is 40 spaces  Values are user defined free text which should contain the following information: - original classification authority name and position or personal identifier if the value in File Classification Authority Type is O; - title of the document or security classification guide used to classify the file if the value in File Classification Authority Type is D; and Derive-Multiple if the file classification was derived from multiple sources. In the latter case, the file originator will maintain a record of the sources used in accordance with existing security directives. One of the multiple sources may also be identified in File Classification Text if desired.
classificationreason	BYTE	File Classification Reason: This field shall contain values indicating the reason for classifying the file. If this field is all spaces, it shall imply that no file classification reason applies.	Default is 1 single space  Valid values are A through G. These correspond to the reasons for original classification per E.O. 12958, Section 1.5.(a) through (g).
classificationsourcedate	STRING	File Security Source Date: This field shall indicate the date of the source used to derive the classification of the file. In the case of multiple sources, the date of the most recent source shall be used. If this field is all spaces, it shall imply that a file security source date does not apply.	Default is 8 spaces  Date Format: CCYYMMDD



Attribute Name	Format	Description	Values
controlnumber	STRING	File Security Control Number: This field shall contain a valid security control number associated with the file. The format of the security control number shall be in accordance with the regulations governing the appropriate security channel(s). If this field is all spaces, it shall imply that no file security control number applies.	Default is 15 spaces
filecopynumber	INT	File Copy Number: This field shall contain the copy number of the file. If this field is all zeros, it shall imply that there is no tracking of file's number of copies.	Default is 00000  Number can range between: 00000 to 99999
numberofcopies	INT	File Number of Copies: This field shall contain the total number of copies of the file. If this field is all zeros, it shall imply that there is no tracking of numbered file copies.	Default is 00000  Number can range between: 00000 to 99999

### 5.6.2.1.1.3 JPEG 2000 XML Example

**Table 5-14: XML Tags for the JPEG 2000 Metadata**

```
<?xml version="1.0" encoding="UTF-8"?>
<JP2METADATA name="JPEG2000XML"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="JP2MetaData.xsd">

  <ORIGIN>
    <datetime></datetime>
    <originatingstationid></originatingstationid>
    <originatorname></originatorname>
    <originatorphone></originatorphone>
    <originatororganization></originatororganization>
    <originatoraddress></originatoraddress>
    <originatoremail></originatoremail>
    <originatorwebsite></originatorwebsite>
    <originatorremark></originatorremark>
  </ORIGIN>
  <SECURITY>
    <classificationlevel></classificationlevel>
    <system></system>
    <codewords></codewords>
    <controlhandling></controlhandling>
    <releaseinstructions></releaseinstructions>
    <declassificationtype></declassificationtype>
    <declassificationdate></declassificationdate>
    <declassificationexemption></declassificationexemption>
    <filedowngrade></filedowngrade>
    <filedowngradedate></filedowngradedate>
    <classificationtext></classificationtext>
    <classificationauthoritytype></classificationauthoritytype>
    <classificationauthority></classificationauthority>
    <classificationreason></classificationreason>
    <securitysourcedate></securitysourcedate>
    <controlnumber></controlnumber>
    <filecopynumber></filecopynumber>
    <numberofcopies></numberofcopies>
  </SECURITY>
</JP2METADATA>
```

### 5.6.2.2 List of all Imagery Dataset Components

The Imagery Dataset is comprised of several components listed here and detailed in the subsequent sections.



Table 5-15: Imagery Dataset Components

CS1	CS2	File Extension	Component Name	Component Description
Dataset 004, Imagery				
001	001	*.jp2	Yearly VSTI Representation	Corresponds to the terrain imagery draped (orthographically) over the terrain skin derived from the Primary Terrain Elevation Dataset. This is the preferred Dataset Component for year-round representative terrain imagery. It may be single-channel monochrome or 3-channel color image. This Dataset Component follows the center grid conventions. Can be used interchangeably with all other Alternate VSTI representations.
002	001..004	*.jp2	Seasonal VSTI Representations	Deprecated – Replaced with Quarterly VSTI Representations below
003	001..012	*.jp2	Monthly VSTI Representations	Monthly equivalent of Yearly VSTI representation, i.e., this is the preferred Dataset Component for month-based representative terrain imagery. Can be used interchangeably with all other Alternate VSTI representations.
004	001..004	*.jp2	Quarterly VSTI Representations	Equivalent to Yearly VSTI representation but for the selected quarter of the year. Can be used interchangeably with all other Alternate VSTI representations.
005	001	*.jp2	Subordinate VSTLM	Corresponds to the terrain light maps draped (orthographically) over the terrain skin derived from the Primary Terrain Elevation Dataset. It may be single-channel monochrome or 3-channel color image. This Dataset Component follows the center grid conventions.

### 5.6.2.3 Visible Spectrum Terrain Imagery (VSTI) Components

The VSTI component provides the visible spectrum imagery that is geo-graphically draped (and usually ortho-rectified) over the geometric representation of the terrain skin that is stored in the Primary Terrain Elevation Dataset. The CDB Specification provides the means to (optionally) store alternate representations of the terrain imagery in order to provide the simulation client-devices terrain representations that best represent the time-of-year being simulated. There are three alternate approaches to the generation and storage of the VSTI Imagery Dataset and they are organized as follows:

1. **Yearly:** The first approach requires a single, year-round representation of the terrain imagery.



2. **Quarterly:** The second approach requires four variants of the terrain imagery, one per calendar-year quarter<sup>63</sup>.
3. **Monthly:** The third approach requires monthly-variants of the terrain imagery, one per month.

The VSTI Imagery Datasets can be provided and stored in any combination, be it yearly, quarterly and/or monthly.

The VSTI dataset implicitly follows the center grid element conventions.

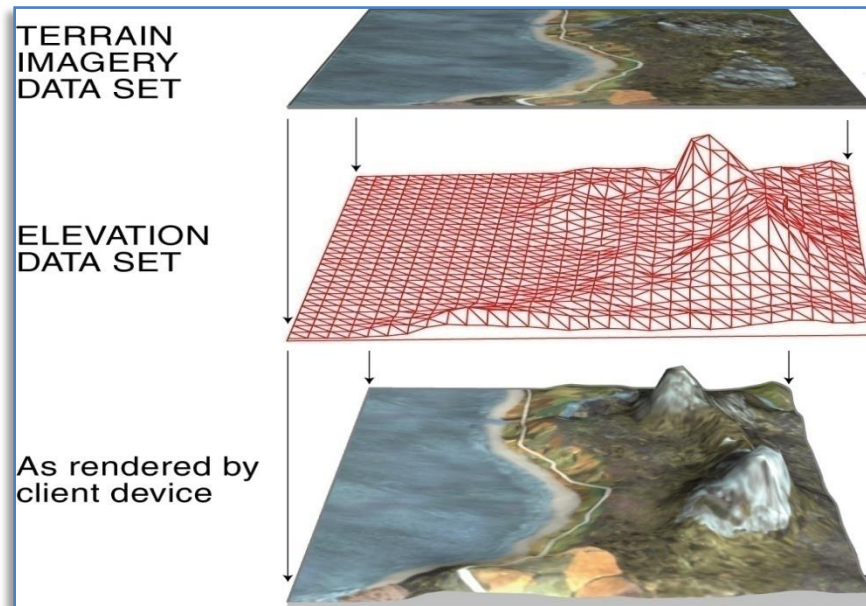


Figure 5-25: Projection of Terrain Imagery Dataset onto Terrain Elevation Dataset

The CDB grid representation of this raster imagery assumes a gamma of 1.0 (see Appendix G) and a color space model in conformance with Windows sRGB or YUV Color Space Profile. sRGB is the default color space in Windows, based on the IEC 61966-2-1 Standard. CDB terrain imagery can optionally be compressed into JPEG 2000 with varying degrees of quantization (quality) levels. However, if using a quantization level different than 0, lossy image results in possible image degradation and artifact addition.

#### 5.6.2.3.1 Data Type

The VSTI component is represented as single-channel gray-scale images, or as triple-channel non-palettred color images in JPEG 2000 format. The use of transparency on terrain imagery is not allowed.

<sup>63</sup> Each quarter corresponds to specific months of the year. This concept of calendar-year quarters is distinct from the concept of seasons whereby the later depends on whether the user is in the northern or the southern hemisphere of Earth.



### 5.6.2.3.2 Default Read Value

Simulator client-devices should default the VSTI values if the data values are not available (files associated with the VSTI dataset for the area covered by a tile, at a given LOD or coarser, are either missing or cannot be accessed). The default value can be found in \CDB\Metadata\Defaults.xml and can be provided to the client-devices on demand. In the case where the default value cannot be found, the CDB Specification recommends that client-devices use a default value of half-intensity (0.5). Note that the default values are expressed as floating-point numbers ranging from 0.0 to 1.0. This ensures that the default is interpreted in a consistent manner independently of the data representation in the \*.jp2 file.

Simulation client-devices are required to select the VSTI texture that best represents the simulation date. The retrieval of VSTI textures by the client-devices must follow the following conventions:

1. The simulation date is converted to a month of the year
2. If the monthly VSTI representation for that month number is absent, then the client-device is required to determine which quarter of the year it is and search for the quarterly representation of the VSTI
3. If a quarterly representation is absent, then the client-device is required to search for a yearly representation of the VSTI
4. if the yearly representation is absent, then the client-device is required to default to the Yearly default values found in \CDB\Metadata\Defaults.xml as follows:
  - a. Default\_VSTI\_Y\_Mono
  - b. Default\_VSTI\_Y\_Red
  - c. Default\_VSTI\_Y\_Green
  - d. Default\_VSTI\_Y\_Blue

The above conventions are summarized in Table 5-16.

Table 5-16: VSTI Default Read Values

Monthly		Quarterly	Yearly	Default
January	001	001	001	Default_VSTI_Y_Mono Default_VSTI_Y_Red Default_VSTI_Y_Green Default_VSTI_Y_Blue
February	002			
March	003			
April	004	002		
May	005			
June	006			
July	007	003		
August	008			
September	009			

Monthly		Quarterly	Yearly	Default
October	010	004		
November	011			
December	012			

#### 5.6.2.3.3 Default Gamma Correction

The default gamma correction is defined by Default\_Imagery\_Gamma found in the Defaults.xml metadata file. If Default\_Imagery\_Gamma is not found in Defaults.xml, or if Defaults.xml is not found in the metadata directory, assume a default gamma correction of 1.0.

#### 5.6.2.3.4 Default Write Value

The files associated with the VSTI component for area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.

### 5.6.2.4 Visible Spectrum Terrain Light Map (VSTLM) Component

The VSTLM component provides the visible spectrum terrain light maps that are orthographically draped over the terrain skin (e.g., Primary Terrain Elevation Dataset) and onto T2DModels. In addition, client-devices can also use the VSTLM component to orthographically project the light map onto GTModels, GSModels and statically-positioned MModels.

Light maps fall under the category of subordinate textures. The light maps are used in low illumination conditions (dusk, dawn, night) to represent the combined illumination effect of man-made light sources (primarily lamp-posts) on the terrain. The technique provides a convenient means to produce interesting and entirely predictable lighting effects without resorting to computationally intensive local light sources.

The light map adds to the lighting levels provided by the simulated ambient light level; the combined ambient lighting and the light map together modulate the underlying VSTI. Light maps can be created in a number of ways, either manually with a tool such as Photoshop, from night-time imagery or finally from an off-line rendering process that simulates the illumination effect of the urban lighting sources onto the terrain.

#### 5.6.2.4.1 Data Type

The VSTLM component is represented as single-channel gray-scale images, or as triple-channel color images. The data is stored in JPEG 2000 format. Note that in the case of a monochrome VSTLM, the implied chrominance of the VSTLM is white.

#### **5.6.2.4.2 Default Read Value**

Simulator client-devices should default the VSTLM values if the data values are not available (files associated with the VSTLM dataset for the area covered by a tile, at a given LOD or coarser, are either missing or cannot be accessed). The default value can be found in \CDB\Metadata\Defaults.xml and can be provided to the client-devices on demand. In the case where the default value cannot be found, the CDB Specification recommends that client-devices use a default value of zero-intensity (0.0). Note that the default values are expressed as floating-point numbers ranging from 0.0 to 1.0. This ensures that the default is interpreted in a consistent manner independently of its representation in the \*.jp2 file. The default values are:

- Default\_VSTLM\_Mono
- Default\_VSTLM\_Red
- Default\_VSTLM\_Green
- Default\_VSTLM\_Blue

#### **5.6.2.4.3 Default Gamma Correction**

The default gamma correction is defined by Default\_Imagery\_Gamma found in the Defaults.xml metadata file. If Default\_Imagery\_Gamma is not found in Defaults.xml, or if Defaults.xml is not found in the metadata directory, assume a default gamma correction of 1.0.

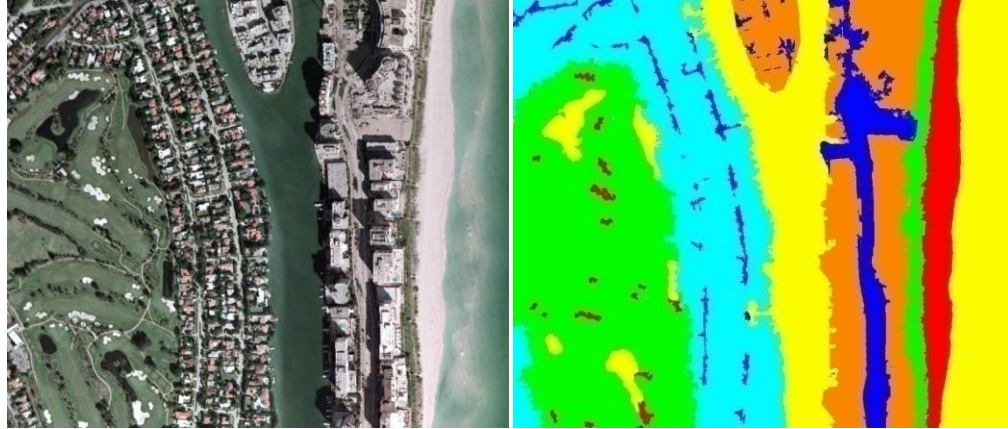
#### **5.6.2.4.4 Default Write Value**

The files associated with the VSTLM component for area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.

### **5.6.3 Tiled Raster Material Dataset**

Historically, Digital Feature Analysis (DFAD) and VPF (Vector Product Format) data have been used to provide the terrain and cultural content information used by the real-time sensors, the computer generated forces and the visual systems. The vectorized outlines of areas tagged with attribution data had a cartoon-like appearance because they did not capture the richly varying mixture of materials. Each geometric shape would be represented as a single material type resulting in simplistic sensor scenes. Sometimes, a locally applied texture pattern would be applied to add some realism to the single material type. While it is still possible to build a CDB in this manner, the preferred approach involves the use of the Raster Material Dataset described here. The Raster Material Dataset can be readily derived from the (image) classification of mono, color or multi-spectral imagery. This Dataset is a material-coded image with mixturing data. It is independent of wavelength (visible, infrared, etc.) and is designed to support geospecific, multi-spectral scene simulation across any computing platform. The Raster Material Dataset is typically generated from material classification and mixturing analysis (see Figure 5-26: Image Classification

Example). It can be developed directly from geospecific imagery, (e.g., SPOT, Landsat) and have a one-to-one correspondence with the image data. The Raster Material Dataset results in a smoothly varying simulation database free of hard edges characteristically found in vectorized DFAD outlines.

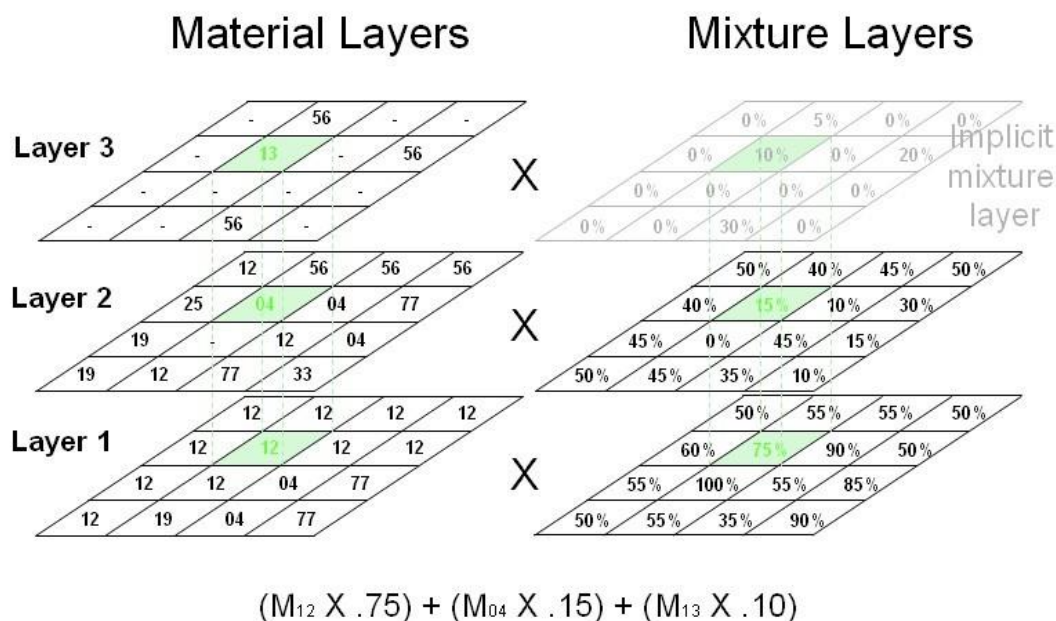


**Figure 5-26: Image Classification Example**

The Raster Material Dataset provides the means to store the types of materials and the area coverage of each material within each pixel of the dataset. In all other aspects, it follows conventions similar to the VSTI dataset.

A Raster Material Dataset consists of a set or stack of “ $n$ ” Material Layers. This stacking arrangement permits the modeler to assign up to “ $n$ ” materials to the area covered by each pixel of the Raster Material Dataset. The Raster Material Dataset also consists of a stack of “ $n-1$ ” mixture layers; the mixture layers define the proportions of materials at each pixel. The CDB Specification makes provision for up to 255 materials, (i.e., any pixel within the Raster Material Dataset can be assigned up to 255 materials).





**Figure 5-27: Example of a Raster Material Dataset**

Figure 5-23: Terrain Elevation, Bathymetry and Tide Components, provides an example of a Raster Material Dataset consisting of 3 material layers and two associated mixture layers. The first Material Layer (i.e., layer 1) consists of a regular grid of pixels; each pixel contains a code that represents the (composite) material with largest area coverage. Likewise, the second Material Layer consists of a grid of pixels whose code represents the (composite) material with second-highest area coverage. Additional layers are added until the area corresponding to the combined area of all (composite) materials at each pixel sums to 100%. Note that in layer 2, the material layer value of some pixels can be ignored (shown as “-” in the illustration) because layer 1 had a material mixture value of 100%. Similarly, the material layer value of some pixels in layer 3 can also be ignored because the mixture layers 1 and 2 add to 100%. In these cases, the CDB Specification recommends that the layer value be assigned a Default\_Material\_Layer value of 0.

**NOTE:** The numeric value for Default\_Material\_Layer is zero (“0”) and is reserved by the CDB Specification.

Mixture Layers represent the percentage area coverage of each material within each pixel of each mixture. Since all layers must add to 100%, it is possible to represent “n” Material Layers with a set of “n-1” Mixture Layers. The last layer is implicit, and it is set to (100% - Sum of areas from previous layers). In the case where there is a single Material Layer, there is no need to store the (implicit) Mixture Layer. When there are two or more Material Layers, the Mixture Layer(s) must be generated.

### 5.6.3.1 List of all Raster Material Dataset Components

The Raster Material Dataset is comprised of several components listed here and detailed in the subsequent sections.

**Table 5-17: Raster Material Dataset Components**

CS1	CS2	File Extension	Component Name	Component Description
Dataset 005, RMTexture				
001	001..255	*.tif	Composite Material Index	Each texel is an index into the Composite Material Table (dataset 006). CS2 is the layer number. Corresponds to a 2D grid of composite material indices draped (orthographically) over the terrain skin derived from the Primary Terrain Elevation Dataset.
002	001..254	*.tif	Composite Material Mixture	Each texel indicates the proportion (between 0.0 and 1.0) of the composite material found in the corresponding material layer. CS2 is the layer number. Corresponds to a 2D grid draped (orthographically) over the terrain skin derived from the Primary Terrain Elevation Dataset. This Dataset component follows the center grid conventions.
Dataset 006, RMDescriptor				
001	001	*.xml	Composite Material Table	The Composite Material Table is referenced by the Composite Material Index component and contains the definitions of the composite materials of a Tile-LOD.

### 5.6.3.2 Composite Material Index Component

As mentioned earlier, the CDB Specification allows pixels of the Raster Material Dataset to consist of several (up to 255) composite materials. To accomplish this, it uses a layering concept that permits the assignment of several composite materials for each pixel in the Material Dataset. As a result, the chosen representation for the Raster Material Dataset consists of a set or stacks of “*n*” Material Layers, where “*n*” is the maximum number of composite materials encountered in any pixel of the CDB tile at the specified LOD.

The code assigned to each pixel of each Material Layer is the index of a Composite Material found in the Terrain Composite Material Table (TCMT) defined in 5.6.3.4.

Each pixel of the first Material Layer (e.g., layer “1”) consists of a code that represents the composite material with largest area coverage for that pixel. Likewise, each pixel of the second Material Layer consists of a code that represents the composite material with second-highest area coverage for that pixel. Additional layers are added until the area corresponding to the combined area of all composite materials at each pixel sums to 100%.



#### **5.6.3.2.1 Data Type**

The Material Layer components are each represented as single-channel, material coded one byte unsigned integer value images stored in TIFF.

#### **5.6.3.2.2 Default Read Value**

If none of the Material Layer components are available (files associated with the Material Layer dataset for the area covered by a tile, at a given LOD or coarser, are either missing or cannot be accessed), simulator client-devices should default to a single Material Layer component whose content defaults to a single default Composite Material. The default Composite Material can be found in \CDB\Metadata\Defaults.xml and can be provided to the client-devices on demand. The default value is:

- Default\_Material\_Layer (0)

In the case where the default value cannot be found, the CDB Specification recommends that client-devices default to single substrate composite material whose base material is:

- Default\_Base\_Material (BM\_LAND-MOOR)

#### **5.6.3.2.3 Default Write Value**

The files associated with the Material Layer components for the area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.

### **5.6.3.3 Composite Material Mixture Component**

A Mixture Layer accompanies each Material Layer; its dimensions are identical to those of the Material Layer. The pixel values of the Mixture Layer “*n*” represent the area coverage of Material Layer “*n*”. Since all layers must add to 100%, it is possible to represent “*n*” Material Layers with a set of “*n-1*” Mixture Layers. As a result, the last layer is implicit, and it is set to (100% - Sum of areas from previous layers). In the case where there is a single Material Layer, there is no need to store the (implicit) Mixture Layer. When there are two or more Material Layers, the Mixture Layer(s) must be generated.

#### **5.6.3.3.1 Data Type**

The Material Mixture components are each stored in a single-channel TIFF file. All values range from 0.0 (0%) to 1.0 (100%). Integral types represent scaled integers to fit the range 0.0 to 1.0; floating-point values are limited to the range 0.0 to 1.0.

#### **5.6.3.3.2 Default Read Value**

If none of the Material Mixture components are available (files associated with the Material Mixture dataset for the area covered by a tile, at a given LOD or coarser, are either missing or cannot be accessed), simulator client-devices should assume equal mixturing for all available Material Layers.

#### **5.6.3.3.3 Default Write Value**

The files associated with the Material Mixture components for the area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.

#### **5.6.3.4 Composite Material Table Component**

This Composite Material Table is called the Terrain CMT, or just TCMT; it provides a list of the Composite Materials shared by the Material Layers of the Material Dataset. There is one TCMT for each CDB tile.

##### **5.6.3.4.1 Data Type**

The TCMT follows the syntax described in Section 2.5.2.2, Composite Material Tables (CMT).

##### **5.6.3.4.2 Default Read Value**

Simulator client-devices should default the Terrain Composite Material Table if file associated with the Terrain Composite Material Table for the area covered by a tile, at a given LOD, is either missing or cannot be accessed. The default values for the Terrain Composite Material Table can be found in \CDB\Metadata\Defaults.xml and can be provided to the client-devices on demand. The default value is a single Composite Material and is named:

- Default\_Material\_Layer (0)

If the default information cannot be found within the \CDB\Metadata\Defaults.xml file, the CDB Specification recommends defaulting to single substrate composite material whose base material is named:

- Default\_Base\_Material (BM\_LAND-MOOR)

If an index is not found in the Terrain Composite Material Table, use the same defaulting mechanism.

##### **5.6.3.4.3 Default Write Value**

The files associated with the Terrain Composite Material Table for the area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.



## 5.7 Tiled Vector Datasets

Vector tiles differ from their raster counterpart in three important ways. First of all, a vector tile's internal structure permits a non-uniform distribution of elements within the tile, (i.e., the position of each element within the tile is explicit). Secondly, the vector tile's internal structure permits a variable number of elements within the tile confines. Finally, it is possible to control the distribution of the element types from a single list.

Conceptually, the LOD of a vector tile implicitly provides the average density of elements within the tile. The run-time level-of-detail behavior that controls the rendered number of data elements depends on various parameters and on the off-line filtering process.

**NOTE:** The LOD referred to in this section concerns itself with the grouping of cultural features into tiles at specified LODs, and not with the geometric accuracy or detail of the modeled representation of these features.

### 5.7.1 Introduction to Vector Datasets

The CDB Specification uses ESRI Shapefiles to represent vector data and attributes. All shape types are supported to represent point, lineal, and areal features.

A point feature is a geographic entity where its simplest representation resolves to a point with general attributes such as size, position, or material. A lineal feature is a geographic entity that defines a one-dimensional feature such as a road, a canal, a river. An areal feature is a geographic entity where its simplest representation resolves to a two-dimension feature with general attributes such as size, position and contours. In this context, a geographic entity is always specified by latitude and longitude coordinates; in turn, the geographic entity is conformed onto the terrain by the client-device.

The lineal and areal feature's representation abstractly resolves to a one or a two dimensional feature. Unless otherwise specifically mentioned in the CDB Specification, lineal and areal feature's representations are not used to model a geometrical representation. However, these features may optionally reference an explicitly modeled representation (for example an OpenFlight model) located in the geospecific model or the geotypical model datasets.

As per ESRI Shapefile Technical Description, the set of attributes of Vector features are stored in dBASE III+ files. Refer to Appendix D for the file format description. The CDB Specification provides three attribution schemas to represent attribution data:

- Instance-level attribution schema
- Class-level attribution schema
- Extended-level attribution schema

To completely represent the vector data and attributes in a given tile, the CDB Specification requires that a Vector dataset consists of some of all of the following files:

- **\*.shp** – feature shape files that provides the geometric aspects of each instance of a vector feature (point, lineal, and areal features). All instances of the feature must be of the same Shape type. While the Shapefile format supports up to 13 different types (each one stored in a different shape file), CDB Specification requires a maximum of one Shapefile type for point features, a maximum of one Shapefile type for lineal features and a maximum of one Shapefile type for areal features for each tile (for a maximum of 3 feature Shapefiles per tile).
- **\*.shx** – feature index files that stores the file offsets and content lengths for each of the records of the feature files. The only purpose of these files is to provide a simple means for clients to step through the individual records of the feature files (i.e., it contains no CDB modeled data).
- **\*.dbf** – feature instance-level files that provide the instance-level attribution data for each of the records of the feature.
- **\*.dbf** – feature class-level files that provide the class-level attribution data for each class of features present in the feature shape files.
- **\*.dbf** – feature extended-level files that provide optional extended-level attribution data for entries in either the feature instance- or class-level files.
- **\*.shp** – figure point shape files allow modelers the ability to assign specific attribution for each point in lineal or areal features. Without this additional Shapefile, the Shapefile format only allows specifying a single attribution for the entire lineal or areal feature. The CDB Specification extends the concept to allow specific attribution to each point of these features while enforcing position correlation. For instance, in case of a PowerLine feature, it is possible to associate, within the same dataset, a different geometric representation of a PowerLine pylon for each point of the lineal and still maintain the relationship between the point and the lineal.
- **\*.shx** – figure point index files that stores the file offsets and content lengths for each of the records of the figure point shape files.
- **\*.dbf** – figure point instance-level files that provide the instance-level attribution data for each of the records of the figure point shape files.
- **\*.dbf** – figure point class-level files that provide the class-level attribution data for each class of features present in the figure point shape files
- **\*.dbf** – figure point extended-level files that provide optional extended-level attribution data for entries in either the figure point instance- or class-level files.



- **\*.dbf** – 2D relationship files. These files establish the relationship of point, lineal, and areal features of a single or different datasets in a tile and between tiles.

In addition to \*.shp, \*.dbf and \*.shx files, the Shapefile standard also refers to a memo file with a \*.dbt file that is used to store comment fields associated with the attribution \*.dbf file.

All of the information that is needed to instance features is organized in accordance to the CDB tile structure. All the tiled Vector dataset files are located in the same directory; the dataset's second component selector (CS2) is used to differentiate between files with the same extension or with the same Vector features. Table 5-18: Component Selector 2 for Vector Dataset, presents the list of codes that are allocated. Note that Vector datasets do not necessarily use all of the Dataset Component Selector 2 reserved codes. Users of the CDB Specification should refer to the appropriate section for an enumeration of the supported File Component Selector 2 codes as well as the ones specific to the Dataset.

The Vector dataset concept and the FACC concepts overlap somewhat; some of the Vector datasets are generalizations or specializations of FACCs. Appendix N provides a recommended mapping of the FACC attributes across the CDB Datasets. Note that the same feature should not have two representations.

**Table 5-18: Component Selector 2 for Vector Datasets**

CS2	File Extension	Dataset Component Name	Supported Shape Type
001	*.shp *.shx *.dbf	Point features	Point, PointZ, PointM, MultiPoint, MultiPointZ, MultiPointM
002	*.dbf	Point feature class-level attributes	N/A
003	*.shp *.shx *.dbf	Lineal features	PolyLine, PolyLineZ, PolyLineM
004	*.dbf	Lineal feature class-level attributes	N/A
005	*.shp *.shx *.dbf	Areal features	Polygon, PolygonZ, PolygonM, Multipatch
006	*.dbf	Areal feature class-level attributes	N/A
007	*.shp *.shx *.dbf	Lineal figure point features	Point, PointZ, PointM, MultiPoint, MultiPointZ, MultiPointM
008	*.dbf	Lineal figure point feature class-level attributes	N/A
009	*.shp *.shx *.dbf	Areal figure point features	Point, PointZ, PointM, MultiPoint, MultiPointZ, MultiPointM
010	*.dbf	Areal figure point feature class-level attributes	N/A
011	*.dbf	2D relationship tile connections	N/A
012		Deprecated	N/A

CS2	File Extension	Dataset Component Name	Supported Shape Type
013		Deprecated	N/A
014		Deprecated	N/A
015	*.dbf	2D relationship dataset connections	N/A
016	*.dbf	Point feature extended-level attributes	N/A
017	*.dbf	Lineal feature extended-level attributes	N/A
018	*.dbf	Areal feature extended-level attributes	N/A
019	*.dbf	Lineal Figure Point extended-level attributes	N/A
020	*.dbf	Areal Figure Point extended-level attributes	N/A

### 5.7.1.1 Shapefile Type Usage and Conventions

This section establishes conventions globally applicable to the usage of all Shapefile features.

#### *For explicitly modeled point cultural features:*

Each point-feature of the CDB can be optionally associated with a GSModel, a GTModel or MModel. The rendering of GSModels, GTModels or MModels by client-devices requires an associated point-feature. The linkage is made through point-feature attributes which together provide the information needed by client-devices to locate the Model from the appropriate Dataset at the appropriate level-of-detail. The following feature attributes provide the necessary linkage:

- **FACC-FSC**: Feature Code and Subcode
- **MODL**: Model Name
- **MODT**: Model Type
- **MLOD**: Model Level-of-Detail
- **MMDC**: Moving Model DIS Code

In the Shapefile, the position of all points is expressed using WGS-84 geographic coordinates (latitude, longitude, altitude), as explained in Appendix K. If the feature has an associated model, client-devices are required to position the model's origin at the specified coordinate, orient the model's Y-axis in accordance to the AO1 attribute, and align the model's Z-axis so that it points up. In the case of Shape types that do not have a Z component value, the object's height value is referenced to the underlying terrain; as a result, client-devices are required to position the model's origin wrt underlying terrain elevation dataset. For Point features with a Z component, client-devices are required to position the model as per the AHGT attribute value. If AHGT is true, the model's origin 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 model's origin is positioned to the value specified by the underlying terrain offset by the Z component value.

***For modeled light points:***

It is common practice within the simulation industry to model light points without their associated support structures. In this case, the preferred way to model light points is through the use of point-features within the Airport and Environmental Light-Point Features Datasets of the CDB; consequently, there are no Models associated with Airport and Environmental Light-Point Features.

Note however that is entirely permissible to also model lights points with their associated support structures. In this case, the CDB OpenFlight Model representing the support structure also contains light points as specified in section 6.11, Model Light Points.

The “modeling” of light points is accomplished via the following light-point feature attributes:

- ***LTYP***: Light Type
- ***LPH***: Light Phase
- ***AOI***: Angle of Orientation

The position of light points are expressed using WGS-84 geographic coordinates (latitude, longitude, altitude), as explained in Appendix K. Client-devices are required to position the center of the light point at the specified coordinate, orient directional light points in accordance to the AO1 attribute. The elevation angle component of a directional light point is intrinsic to its type (for instance a VASI\TypeT\2.5\_Degree\Fly-Up1\_light should be used to represent a Type VASI light used for a 2.5 degree glide slope). In the case of Shape types that do not have a Z component value, the light point’s height value is referenced to the underlying terrain; as a result, client-devices are required to elevate the light point’s center wrt underlying terrain elevation dataset. For Light Point features with a Z component, client-devices are required to position the light point’s center as per the AHGT value. 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. 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.

***For point, lineal and areal features that are not modeled:***

The CDB data model does not make mandatory that all features of the CDB be modeled; as a result, each feature is ***optionally*** associated with a GSModel, a GTModel or a MModel.

The position of vertices is expressed using WGS-84 geographic coordinates (latitude, longitude, altitude), as explained in Appendix K. 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; as a result, client-devices are required to position the vertex’s origin wrt underlying terrain elevation dataset. For Shape types with a Z component,



client-devices are required to 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.

AHGT attribute, when present, is always ignored when the Z component value does not exist.

The bounding box coordinates Xmin, Ymin, Xmax, Ymax required by some Shape types are expressed using WGS-84 geographic coordinates (in accordance to Appendix K).

The value of M and Mrange found in some of the Shape types (PointM, MultiPointM, PolygonM, and PolyLineM) is ignored by client-devices.

#### **5.7.1.1.1 Notes about Shapefile Polygon Shapes**

Even though the Shapefile standard is very versatile, it also enforces some guidelines with respect to the Polygon Shapes. Those guidelines are referred to in Appendix D – Shapefile July 1998 Technical Description – Annotated.

The key aspects that should be respected while generating Polygon Shapes are re-listed below:

- Has no self-intersections or co-linear segments
- Has no identical consecutive points (no zero-length segments)
- Does not degenerate into zero-area parts
- Does not have clock-wise inner rings (“Dirty Polygon”)

Although the above are guidelines, Shapefile readers shall handle such cases with proper error handling and reporting.

#### **5.7.1.2 CDB Attribution**

Attributes are used to describe one or more real or virtual characteristics of a feature. Features can be assigned a variable number of attributes.

##### **5.7.1.2.1 Attribute Code**

A unique four-digit numeric code is associated to each attribute. For example, the attribute "Angle of Orientation" has an attribute code of “0003”.

##### **5.7.1.2.2 Attribute Identifier**

A unique three-character or four-character alphanumeric identifier is associated to the attributes that are governed by this Specification. Attributes other than those governed by the CDB Specification may not have an assigned identifier. For example, the CDB attribute "Length" has the “LEN” identifier. The identifier is a

case-sensitive string of up to 10 characters. In the case of instance-level and class-level attributes, the identifier is used as the name of the \*.dbf column.

#### 5.7.1.2.3 Attribute Semantics

Each attribute is associated with a textual description (describing semantic information), which provides a human readable definition of the attribute.

#### 5.7.1.2.4 Attributes Values

A value can be assigned to each attribute. The data type, length, format, range, usage, units, compatibility and schema of each attribute value is governed by this Specification. Attribute values give quantitative/qualitative meaning to the attribute.

#### 5.7.1.2.5 Attribute Usage

CDB attribution usage falls in the following categories:

**Mandatory:** A mandatory attribute is an attribute whose value must be provided for all of the features of a specified dataset, i.e., a producer of CDB data (e.g., tools) is required to generate values for mandatory attributes. Consumers of CDB data (tools and/or simulator client-devices) can rely on the availability of mandatory attributes. A CDB with missing mandatory attributes is considered non-compliant by this Specification.

**Recommended:** A recommended attribute is an attribute whose value should be provided for all of the features of a specified dataset. Consumers of CDB data (tools and/or simulator client-devices) can always rely on the availability of recommended attributes since the attribute value is either provided explicitly by the CDB or provided implicitly as a defaulted value in accordance to section 5.7.1.3, CDB Attributes. A CDB with defaulted recommended attributes is considered compliant by this Specification; however, the performance of one or more of the client-devices (commonly found on simulation devices) may be adversely affected.

**Optional:** An optional attribute is an attribute whose value may (optionally) be provided for all of the features of a specified dataset. Consumers of CDB data (tools and/or simulator client-devices) cannot rely on the availability of optional attributes. A CDB with missing optional attributes is considered compliant by this Specification; however, the performance of one or more of the client-devices (commonly found on simulation devices) may be enhanced by including the optional attributes.

**Dependent:** A dependent attribute is an attribute whose value depends on another attribute, be it mandatory, recommended, or optional. The attribute is considered mandatory if the attribute it depends on is mandatory. Likewise, the attribute is considered recommended if the attribute it depends on is recommended. Finally, the attribute is considered optional if the attribute it depends on is optional.

Note that attribute usage information for each of the CDB attributes can be found in section 5.7.1.3, CDB Attributes and in Table 5-27: Allocation of CDB Attributes to Vector Datasets

#### **5.7.1.2.6 Attribution Data Compatibility**

The CDB Specification provides a flexible means to tag features with attribution data. The CDB Specification accommodates the vast majority of attribution data that is in use today and available through formats and products supported by the NGA and other US governmental agencies. The CDB Specification provides the means to attribute features with attribution data with varied origins.

##### **5.7.1.2.6.1 CDB Attributes**

CDB attributes are attributes whose semantics, data type, length, format, range, usage, units, compatibility and schema are entirely governed by the CDB specification. Most of these attributes are unique to the CDB Specification, i.e., these attributes are not found in source data that conforms to various (US) governmental standards and Specifications. As a result, this attribution data must be derived via CDB tool automation or provided directly by the user.

##### **5.7.1.2.6.2 Geomatics Attributes**

Geomatics attributes are attributes whose semantics, data type, length, format, range, usage, and units, are governed by various governmental/industrial Specifications and standards. Such attributes are generally found in source data that conforms to such standards and specifications. While the CDB Specification itself does not define and govern the usage of these attributes, it nonetheless accommodates their storage within the repository structure of the CDB.

##### **5.7.1.2.6.3 Vendor Attributes**

Vendor attributes are attributes whose semantics, data type, length, format, range, usage, and units are governed by one or more vendors. In general, such attributes cannot be used by other vendors since they are often proprietary. Such attributes exclude the above two types of attributes and are generally unique to each vendor. While the CDB Specification itself does not define and govern the usage of these attributes, it nonetheless accommodates their storage within a CDB.

#### **5.7.1.2.7 Attribution Schemas**

The CDB Specification offers three different attribution schemas. Each of the schemas offers different trade-offs in the manner attribution data is accessed and stored. Each of these schemas is largely motivated by the storage size considerations, and flexibility in the manner attribution data can be assigned to individual features and to groups of features.



The three attribution schemas supported by the CDB Specification are:

- Instance-level schema
- Class-level schema
- Extended-level schema

The preferred attribution schema is Instance-level. This choice reduces the number of files that must be read by client-devices and, thus, improves their runtime performance.

#### 5.7.1.2.7.1 Instance-level Schema

This is the preferred attribution schema that can be used with all CDB attributes allowed for a given type of features. This is the only attribution schema that can be used with features whose attributes and attribute values vary with each instance of a feature in a dataset. The attributes and their values are specified as attribution columns in the instance-level \*.dbf file that accompanies the dataset's \*.shp file. This \*.dbf file is referred to as the Dataset Instance-level\*.dbf file.

Each instance of a feature is characterized by a corresponding set of instance-level attributes implemented as a row within the instance-level \*.dbf file. Each attribute is uniquely defined by an attribute identifier. Each row of this instance-level \*.dbf file contains the instance-level attribute values for a corresponding feature in the \*.shp file. The first column of each row within the instance-level \*.dbf is always the classname (CNAM). If the classname is not used, its value is set to blank, and all of the classname attributes must be added to the instance-level \*.dbf file. The number of columns in a Dataset Instance-level \*.dbf file is different for each dataset. All of the instance-level attributes are CDB attributes.

Dataset *.shp File		Dataset Instance-level *.dbf								
		CNAM	A01	LENL	LPN	RTAI	SCALX	SCALY	SCALZ	::
record 1	→	-	5.2°	1000 m	1	89%	1.0	1.0	1.0	...
record 2	→	-	15.0°	2500 m	1	71%	1.5	1.5	1.5	...
record 3	→	-	90.0°	565 m	1	99%	1.0	1.0	1.0	...
record 4	→	-	82.0°	1003 m	1	85%	1.0	1.0	1.0	...
.	→	-	.	.	.	.	.	.	.	...
.	→	-	.	.	.	.	.	.	.	...
.	→	-	.	.	.	.	.	.	.	...

Figure 5-28: Instance-level Attribution Schema

#### 5.7.1.2.7.2 Class-level Schema

This attribution schema can be used for features whose attributes and attribute values can be shared by one or more of the instances of a feature in a dataset.

The attributes and their values are logically re-grouped under a classname (CNAM attribute) that stands for the group of attributes specific to that class. Each row of the class-level \*.dbf file corresponds to a classname found in the instance-level \*.dbf shape file. Each attribute class is characterized by a set of attributes implemented as a row within the class-level \*.dbf file. Each attribute is uniquely defined by an attribute identifier. The first column of the file is the classname and acts as the primary key to access table entries; all other rows correspond to the attributes represented by the classname. All of the class-level attributes are CDB attributes.

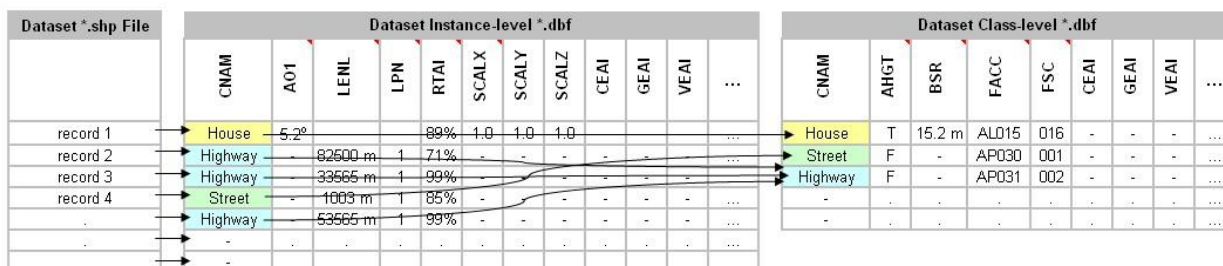


Figure 5-29: Class-level Attribution Schema

Eventhough Class-level schema can be seen as a mean to group common CDB attributes into a single file, it is recommended to store them all using Instance-level schema. Doing so reduces the number of files that must be read by client-devices and, thus, improves their runtime performance.

#### 5.7.1.2.7.3 Extended-level Schema

The CDB Specification provides an alternate attribution schema that can be used (in many cases) to supplement the instance-level and class-level schemas.

The extended-level schema can be used to represent CDB attributes, Geomatics attributes and Vendor attributes. However, the extended-level schema is the only means by which Geomatics attributes and Vendor attributes can be accessed.

Linkage to the extended-level CDB attribution data is accomplished through the CEAI attribute; CEAI is an index to a link list of CDB attributes stored in the extended-level \*.dbf file. Similarly, the GEAI and VEAI attributes are also indices to link lists of attributes stored in the extended-level \*.dbf file.

#### 5.7.1.2.7.4 Structure of Extended-level dbf Files

Each row of the Extended-Level \*.dbf files correspond to an attribute. Each attribute row consists of four columns as follows:

Column 1 – **LNK (Link)**: The first column is a numeric 6-digit index to the next entry of a link list of attributes (a value of 0 marks the end of the list). The LNK field provides a means to organize attributes into link lists of attributes that in turn can be associated with a feature.

Column 2 – **GRP (Group)**: The second column provides an 8-character string that is used to name the group to which the extended attributes belongs to. The actual value of this character string is arbitrary and provides an indication of the source of the attribute. In practice, attributes belongs to one of three (3) groups: CDB, Geomatics, and Vendor. If the extended-level attribute is one of the CDB attributes of section 5.7.1.2.7.5, the group name is “CDB”. If the extended-level attribute belongs to one of the Geomatics standards (such as “DIGEST”, “VMAP”, “SEDRIS”, “DGIWG”, “UHRB”), it is recommended to use the name of the standard as the group name. If the extended-level attribute is a vendor-specific attribute, then the group name should represent the name of the vendor (such as “CAE-M”, “Presagis”, “Thales”, “Rockwell”).

Column 3 – **EAC (Environment Attribute Code)**: The third column provides a unique four-digit numeric code for each attribute type. The codes for the CDB attributes can be found in section 5.7.1.3, CDB Attributes. Note however, that the codes for the Geomatics and Vendor attributes are not specified by this Specification. Instead, this Specification provides a metadata schema that allows developers to describe these attributes. See section 5.1.7, CDB Attributes Metadata, for details.

Column 4 – **EAV (Environment Attribute Value)**: The fourth column provides a data value for the attribute. The data value is represented by general-purpose 16-character alphanumeric string. In the case where more than 16-characters are needed to represent a data value, the remaining characters are provided by appending consecutive row(s) with the same GRP and EAC values; the value of LNK is incremented for each of the consecutive row(s). The interpretation of the data value is governed by metadata that describes the data type, the data format, the allowable range of the data, the numerical precision of the data, the units associated with the data, etc for each attribute type.

#### **5.7.1.2.7.5 Example**

The following example illustrates the relations between Shapefiles and dBASE files where instance, class, and extended-level attributes are stored. The example focuses on extended-level attributes. Note that it is possible to extend the list of instance and class attributes through the use of the CEAI, GEAI, and VEAI attributes.

The attributes associated with the instance of Shape 2 are extended with CDB attributes because CEAI has the value 4; that is an index into the Extended-level attributes dBASE file, it points to record 4. By following the link (LNK) in each record, the complete list of extended attributes contains records 4, 5, and 8. These records add 3 CDB attributes: 5, 54, and 25. These codes respectively correspond to APID, RWID, and GAID. Their respective values are Airport CYUL, Runway 06L, and Gate B23.

The attributes that belong to the “Container” class are also extended with CDB attributes as indicated by the value 6 of the CEA attribute. Record 6 adds CDB attribute 29, LACC, with a value of 1; record 7 adds CDB attribute 60, SSC, with a value of 84.

The attributes of the “Railroad” class are extended by Geomatics attributes as indicated by GEAI and its value of 1. This adds 3 DIGEST geomatics attributes (numbered 1, 2 and 3) that are defined in Geomatics\_Attributes.xml.

Finally, the “Highway” class attributes are extended with a single vendor attribute stored in record 9 and 10 (VEAI points to record 9 which points to record 10). The client detects that this is a single attribute (and not two separate attributes) because the two records have identical values for their GRP and EAC attributes. The vendor is identified as “ABC Inc.”; attribute 1, defined in Vendor\_Attributes.xml, has the value “1234567890ABCDEFGHIJ”.





Shapefile (*.shp)	Instance-Level Attributes (*.dbf)											
	CNAM	AO1	LENL	LPN	RTAI	SCALX	SCALY	SCALZ	CEAI	GEAI	VEAI	..
Shape 1	House	5.2°	-	-	89%	1.0	1.0	1.0	-	-	-	
Shape 2	Highway	-	82 500 m	1	71%	-	-	-	4	-	-	
Shape 3	Highway	-	33 565 m	1	99%	-	-	-	-	-	-	
Shape 4	Railroad	-	154 000 m	1	85%	-	-	-	-	-	-	
Shape 5	Highway	-	53565 m	1	99%	-	-	-	-	-	-	
...	...											

Class-Level Attributes (*.dbf)							
Header	CNAM	BSR	FACC	FSC	CEAI	GEAI	VEAI
Record 1	Container	15.2 m	VX000	000	6	-	-
Record 2	Railroad	-	AN010	000	1	-	-
Record 3	Highway	-	AP030	002	-	-	9
Record 4	...						

Extended-Level Attributes (*.dbf)				
Header	LNK	GRP	FAC	EAV
Record 1	2	DIGEST	1	1
Record 2	3	DIGEST	2	6.45
Record 3	0	DIGEST	3	XYZ...
Record 4	5	CDB	5	CYUL
Record 5	8	CDB	54	06L
Record 6	7	CDB	29	1
Record 7	0	CDB	60	84
Record 8	0	CDB	25	B23
Record 9	10	ABC Inc.	1	1234567890ABCDEF
Record 10	0	ABC Inc.	1	GHIJ

Figure 5-30: Relation between Shapes and Attributes

Note that it is possible to simultaneously extend a record (instance and class) with CDB, Geomatics, and vendor attributes. The example does not illustrate this situation to keep it (relatively) simple.

### 5.7.1.3 CDB Attributes

This section provides a list and description of the attributes that are governed by the CDB Specification. Note that it is possible to provide attributes other than those listed here by making use of the Geomatics and Vendor Extended-level attribution schema.

#### 5.7.1.3.1 ATARS Extended Attribute Code (AEAC) – Deprecated

Description: A unique numeric identifier that points to the entry number of the ATARS Extended Attribution file for the current dataset. This entry is provided for legacy database generation facility considerations only; CDB-compliant devices are not required to read and interpret this field. The ATARS Extended Attribution file should be located in the same directory as the instance-level attribution file. An empty AEAC field (i.e., null string) is allowed.

Identifier: AEAC

Code: 0001

Data Type: numeric

Length: 9 characters

Format: integer

Range: 0 to 999,999,999

Usage Note: Optional. Use when ATARS extended attribution is required.

Unit: N/A

Compatibility: CDB 3.0

#### 5.7.1.3.2 Absolute Height Flag (AHGT)

Description: Indicates how to interpret the Z component of a vertex.

Identifier: AHGT

Code: 0002

Data Type: Logical

Length: 1 character

Format: N/A

Range: F, f, N, n (false) and T, t, Y, y (true)

Usage Note: Optional. Specifies how to interpret shape type features with a Z component. If AHGT is true, the feature is positioned to the value specified by the Z component (Absolute Terrain Altitude), irrelevant of the terrain elevation dataset. If AHGT is false or not present, the

feature is positioned to the value specified by the underlying terrain offset by the Z component value. Refer to section 5.7.1.1, Shapefile Type Usage and Conventions for more details. AHGT can be present only in datasets using PointZ, PolylineZ, PolygonZ and MultiPointZ Shape types. AHGT should not be present for all other Shape types or must be ignored otherwise. Refer to Appendix A – “How to Interpret the AHGT, HGT, BSR, BBH, and Z Attributes” for additional usage guidelines.

Unit: N/A  
Default: False  
Compatibility: CDB 3.0

**NOTE:** It is recommended that the AHGT flag be set to false because it facilitates the creation of CDB datasets that are independent of each others. When the Z coordinate (altitude) of a feature is relative to the ground, the terrain elevation dataset can be updated without the need to recompute the altitude of the feature.

**CAUTION:** When the AHGT flag is set to true, the feature will be at a fixed WGS-84 elevation independently of the terrain LOD selected by the client-device. As a result, there is no guarantee that the feature (and its modeled representation) will remain above the terrain across all terrain LODs.

**RECOMMENDATION:** Limit the use of AHGT=TRUE to data whose source is inherently absolute. Such source data include geodetic marks or survey marks that provide a known position in terms of latitude, longitude, and altitude. Good examples of such markers are boundary markers between countries.

### 5.7.1.3.3 Angle of Orientation (AO1)

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

Identifier: AO1

Code: 0003

Data Type: numeric

Length: 7 characters

Format:	floating-point (recommended precision of 3.3)
Range:	0.000 to 360.000
Usage Note:	Recommended. CDB readers should default to a value of 0.000 if AO1 is missing. Applicable to Point, Light Point, Moving Model Location and Figure Point features. When used in conjunction with the PowerLineNetwork dataset, AO1 corresponds to the orientation of the Y-axis of the modeled pylon. The modeled pylon should be oriented (in its local Cartesian space) so that the wires nominally attach along the Y-axis. Refer to Appendix A – “Creating a 3D Model for a Powerline Pylon” for additional usage guidelines.
Unit:	degree
Default:	0.000
Compatibility:	CDB 3.0, DIGEST v2.1

#### **5.7.1.3.4 Airport Feature Name (APFN) – Deprecated**

Description:	This name is used to distinguish and categorize features within the list of available Airport Lineal features and Airport Areal features.
Identifier:	APFN
Code:	0004
Data Type:	text
Length:	24 characters
Format:	Alpha characters
Range:	N/A
Usage Note:	N/A
Unit:	N/A
Default:	N/A
Compatibility:	CDB 3.0

#### **5.7.1.3.5 Airport ID (APID)**

Description:	A unique alphanumeric identifier that points to a record in the NavData Airport or Heliport dataset (i.e., a link to the Airport or the Heliport description in the NavData dataset). This ID is the value of the field Ident of the Airport or Heliport dataset. Note that all of the lights located in vector datasets that are associated with the operation of an airport (including runway lights and lighting systems) are required to reference an airport or heliport in the NavData dataset. All man-made features associated with an airport or heliport must be
--------------	--



assigned an APID attribute; the APID attribute is not required for features unrelated to airports or heliports.

Identifier:	APID
Code:	0005
Data Type:	alphanumeric
Length:	6 characters
Format:	N/A
Range:	N/A
Usage Note:	Recommended for all Airport Light Points and airport-related T2DModels (such as runway/taxiway/apron surfaces, and markings) Failure to appropriately tag airport culture with APID attribute will result in reduced control of airport-related culture by simulator. Optional for Location Points, Environmental Light Points, and Moving Model Location features that fall within the confines of an airport and for which control of the feature is desirable.
Unit:	N/A
Default:	None
Compatibility:	CDB 3.0

#### 5.7.1.3.6 Bounding Box Height (BBH)

#### 5.7.1.3.7 Bounding Box Width (BBW)

#### 5.7.1.3.8 Bounding Box Length (BBL)

Description:	The Height/Width/Length of the Bounding Box of the 3D model associated with a point feature. It is the dimension of the box centered at the model origin and that bounds the portion of the model above its XY plane, including the envelopes of all articulated parts. BBH refers to height of the box above the XY plane of the model, BBW refers to the width of the box along the X-axis, and BBL refers to the length of the box along the Y-axis. Note that for 3D models used as cultural features, the XY plane of the model corresponds to its ground reference plane. The value of BBH, BBW and BBL should be accounted for by client-devices (in combination with other information) to determine the appropriate distance at which the model should be paged-in, rendered or processed. BBH, BBW and BBL are usually generated through database authoring tool automation.
Identifiers:	BBH, BBW, BBL
Codes:	0006, 0007, 0008

Data Type: numeric  
Length: 9 characters  
Format: floating-point (recommended precision 5.3)  
Range: 0.000 to 999999.999  
Usage Note: Optional on features for which a MODL has been assigned. The dimension of the bounding box is intrinsic to the model and identical for all LOD representations. Refer to Appendix A – “How to Interpret the AHGT, HGT, BSR, BBH, and Z Attributes” for additional usage guidelines.  
Unit: meters  
Default: BBH defaults to the value of BSR  
BBW and BBL default to twice the value of BSR  
Compatibility: CDB 3.0

#### 5.7.1.3.9 Boundary Type (BOTY)

Description: A value that uniquely attributes a boundary according to the enumerators found here.  
Identifier: BOTY  
Code: 0009  
Data Type: Enumeration per Table 5-19: Boundary Type Enumeration Values  
Length: 3 characters  
Format: integer  
Range: 0 to 999  
Usage Note: Optional. See table below for a list of accepted values. Can be used only with Boundary Point, Lineal or Areal Feature Datasets (which are part of the Geopolitical Datasets)  
Unit: N/A  
Default: 0  
Compatibility: CDB 3.0

**Table 5-19: Boundary Type Enumeration Values**

BOTY Code	Description
0	Unknown
1	Continental
2	International



3	Interstate
4	Inter-provincial
5	Territorial
6	Economic
7	Regional
8	Communal
9	Tourist
10	Private Zone
11	Military District
12	Disputed
13	Populated Place
14	Non-capital City
15	Time Zone Delimiter
16	International Date Line
17	Capital City
997	Unpopulated
998	Not Applicable
999	Other

#### **5.7.1.3.10 Bounding Sphere Radius (BSR)**

**Description:** The radius of a feature. In the case where a feature references an associated 3D model, it is the radius of the hemisphere centered at the model origin and that bounds the portion of the model above its XY plane, including the envelopes of all articulated parts. Note that for 3D models used as cultural features, the XY plane of the model corresponds to its ground reference plane. The value of BSR should be accounted for by client-devices (in combination with other information) to determine the appropriate distance at which the model should be paged-in, rendered or processed. When the feature does not reference a 3D model, BSR is the radius of the abstract point representing the feature (e.g., a city).

**Identifier:** BSR

**Code:** 0010

**Data Type:** numeric

**Length:** 9 characters

**Format:** floating-point (recommended precision 5.3)

**Range:** 0.000 to 99,999.999



Usage Note: Mandatory for features for which a MODL has been assigned, but optional for geopolitical point features. The dimension of the bounding sphere is intrinsic to the model and identical for all LOD representations. Refer to Appendix A – “How to Interpret the AHGT, HGT, BSR, BBH, and Z Attributes” for additional usage guidelines.

Unit: meters

Default: None

Compatibility: CDB 3.0

#### 5.7.1.3.11 CDB Extended Attribute Index (CEAI)

Description: An index that points to a row entry of a CDB Extended Attribution file for the current dataset. This entry permits users to store an index to a link list set of CDB-specific attributes. CDB-compliant devices must be capable of reading and interpreting this field. Usage of this attribution is not portable to other simulators because it falls outside of the documented CDB attribution scheme. The CDB Extended Attribution file should be located in the same directory as the instance-level attribution file. An empty CEAI attribute is allowed. Note that the first entry in the CDB Extended Attribution file has an index of 1.

Identifier: CEAI

Code: 0011

Data Type: numeric

Length: 6 characters

Format: integer

Range: 1 to 999,999

Usage Note: Optional. Use when CDB extended attribution is required. A “blank” or a value of 0 indicates that there are no CDB Extended attributes.

Unit: N/A

Default: None

Compatibility: CDB 3.0.

#### 5.7.1.3.12 CDB Extended Attribute Code (CEAC) – Deprecated

Description: A unique numeric identifier that points to the entry number of a CDB Extended Attribution file for the current dataset. This entry permits users to store a link to a set of CDB-specific attributes



beyond those explicitly supported by the current version of this Specification. CDB-compliant devices may optionally read and interpret this field due to program requirements that cannot be supported by the current version of the CDB. Usage of this attribution is not portable to other simulators because it falls outside of the documented CDB attribution scheme. The CDB Extended Attribution file should be located in the same directory as the instance-level attribution file. An empty CEAC field is allowed.

Identifier:	CEAC
Code:	0012
Data Type:	numeric
Length:	9 characters
Format:	integer
Range:	0 to 999,999,999
Usage Note:	Optional. Use when CDB extended attribution is required.
Unit:	N/A
Default:	None
Compatibility:	CDB 3.0

### 5.7.1.3.13 Composite Material Index (CMIX)

Description:	Index into the Composite Material Table is used to determine the Base Materials composition of the associated feature. Refer to Section 2.5, Material Naming Conventions for a description on material naming conventions.
Identifier:	CMIX
Code:	0013
Data Type:	numeric
Length:	6 characters
Format:	integer
Range:	0 to 999,999
Usage Note:	Mandatory on most datasets.
Unit:	N/A
Default:	None
Compatibility:	CDB 3.0

#### 5.7.1.3.14 Class Name (CNAM)

Description:	A name that represents the Attribution Class. The class-level attribution schema is described in Section 5.7.1.2.7.2, Class-level Schema. Attributes are referenced via this classname. The classname is used as the primary key to perform searches within the Dataset Class Attribute file.
Identifier:	CNAM
Code:	0014
Data Type:	text
Length:	32 characters
Format:	lexical
Range:	N/A
Usage Note:	Each row of a class-level dBASE file must have a valid CNAM entry; the CNAM must be unique within the file. Each row of an instance-level *.dbf can optionally use the CNAM to refer to class attributes; blank indicates “no class attribute”.
Unit:	N/A
Default:	None
Compatibility:	CDB 3.0

#### 5.7.1.3.15 Damage Level (DAMA)

Description:	Represents the level of damage of the feature and its model, if applicable. The level is expressed as a percentage where a value of 0 means no damage at all and a value of 100 means fully damaged and completely destroyed. In the case of network datasets, the level of damage shall be interpreted as a measure of the incapacity of the feature to perform its function. For instance, a road network whose damage level is 75% tells the client that it is only able to perform 25% of its intended function. As a result, a certain client may decide that it cannot use the road network while another client may continue to do so.
Identifier:	DAMA
Code:	0067
Data Type:	numeric
Length:	3 characters
Format:	integer
Range:	0 to 100



Unit: Percentage

Default: 0

Compatibility: CDB 3.2

Usage Note: In the context of HLA/DIS, the concept of DAMA maps directly to the concepts of Damage State for which the standards define 4 states named No Damage, Slight Damage, Moderate Damage, and Destroyed. The Specification suggests the following mapping between CDB DAMA and HLA/DIS states.

From CDB to HLA/DIS	
DAMA < 25	No Damage
25 ≤ DAMA < 50	Slight Damage
50 ≤ DAMA < 75	Moderate Damage
75 ≤ DAMA	Destroyed

From HLA/DIS to CDB	
No Damage	0
Slight Damage	33
Moderate Damage	66
Destroyed	100

#### **5.7.1.3.16 DIGEST Extended Attribute Code (DEAC) – Deprecated**

Description: A unique numeric identifier that points to the entry number of the DIGEST Extended Attribution file for the current dataset. This entry is provided for legacy database generation facility considerations only; it provides a means for the CDB to act as a repository for legacy DIGEST attribution. CDB-compliant devices are not required to read and interpret this field. The DIGEST Extended Attribution file should be located in the same directory as the instance-level attribution file. An empty DEAC field (i.e., null string) is allowed.

Identifier: DEAC

Code: 0015

Data Type: numeric

Length: 9 characters

Format: integer

Range: 0 to 999,999,999

Usage Note: Optional. Use when DIGEST extended attribution is required.

Unit: N/A

Default: None

Compatibility: CDB 3.0.

#### 5.7.1.3.17 Depth below Surface Level (DEP)

Description:	The depth of a feature. If the feature has no modeled representation, its depth is measured as the distance from the surface level to the lowest point of the feature below the surface <sup>64</sup> . If the feature has an associated 3D model, the depth is measured as the distance from the XY plane of the model to the lowest point of the model below that plane. DEP values are positive numbers.
Identifier:	DEP
Code:	0016
Data Type:	numeric
Length:	9 characters
Format:	floating-point (recommended precision 5.3)
Range:	0.000 to 99999.999
Usage Note:	In the case of ground features, DEP refers to the portion of the feature (or its modeled representation) that is underground. In the case of moving models that are used as geotypical features, DEP refers to the portion of the model that is below the waterline (i.e., the XY plane). In the case of network lineal features such as roads, railroads and powerlines, DEP refers to the depth of the feature under the ground in its vicinity. In the case of hydrographic features, DEP refers to the depth of rivers, lakes, etc <sup>65</sup> . This data is typically used by client-devices that need to determine whether or not a waterway is navigable by ships with a specific draw.
Unit:	meters
Default:	0.000
Compatibility:	CDB 3.0, DIGEST 2.1

#### 5.7.1.3.18 Directivity (DIR)

Description:	The side or sides of a feature that has the greatest reflectivity potential. This data is typically needed for Radar simulation. DIR is used solely for lineal features in accordance to DFAD conventions. If DIR is not equal to 3, then AO1 is the angular distance measured from true north (0 deg) clockwise to the reflective side of the feature.
Identifier:	DIR

---

<sup>64</sup> Surface here refers to the terrain in the immediate vicinity of the feature.

<sup>65</sup> Note, that the CDB has provision for a raster dataset to represent the bathymetry. When provided, the dataset provides a much more detailed underwater profile of hydrographic features.



Code: 0017  
Data Type: numeric  
Length: 3 characters.  
Format: integer. Enumerated per DIGEST  
1: Uni-directional  
2: Bi-directional  
3: Omni-directional  
Range: 0 to 999  
Usage Note: Recommended for lineal features. If absent, client-devices are required to default to a value of 3 – Omni-directional  
Unit: N/A  
Default: 3  
Compatibility: CDB 3.0 and DIGEST

#### 5.7.1.3.19 Density Measure (DML)

Description: Percentage light coverage at night (expressed as a percentage) within the area delimited by an areal feature.  
Identifier: DML  
Code: 0018  
Data Type: numeric  
Length: 3 characters  
Format: integer  
Range: 0 to 100  
Usage Note: Recommended. Applies to Geopolitical Dataset areal features that delineate inhabited areas. If this field is absent, client-devices shall assume 0%.  
Unit: Percentage  
Default: 0  
Compatibility: CDB 3.0

#### 5.7.1.3.20 Density Measure (% roof cover) (DMR)

Description: Roof cover measure by percent within area of feature.  
Identifier: DMR  
Code: 0019

Data Type:	numeric
Length:	3 characters
Format:	integer
Range:	0 to 100
Usage Note:	Recommended for Areal features. If absent, client-devices shall assume 0%.
Unit:	percentage
Default:	0
Compatibility:	CDB 3.0, DIGEST 2.1

#### **5.7.1.3.21 Density Measure (structure count) (DMS)**

Description:	Number of man-made, habitable structures per square kilometer.
Identifier:	DMS
Code:	0020
Data Type:	numeric
Length:	5 characters
Format:	integer
Range:	0 to 99,999 (Note: differs from DIGEST range of -32767 to 32768)
Usage Note:	Recommended for Areal features. If absent, client-devices shall assume 0.
Unit:	N/A
Default:	0
Compatibility:	CDB 3.0, DIGEST 2.1

#### **5.7.1.3.22 Density Measure (% tree/canopy cover) (DMT)**

Description:	Canopy cover measure by percent within area of feature during the summer season.
Identifier:	DMT
Code:	0021
Data Type:	numeric
Length:	3 characters
Format:	integer
Range:	0 to 100





Usage Note: Recommended for Areal features. If absent, client-devices shall assume 0%.

Unit: percentage

Default: 0

Compatibility: CDB 3.0, DIGEST 2.1

### 5.7.1.3.23 End Junction ID (EJID)

Description: A Junction Identification Number that is used to virtually connect the end point of a lineal to another point, lineal or areal feature. Lineal features stored in the same shape file having the same SJID or EJID are connected. Lineal features stored in different shape files having the same SJID or EJID as the JID listed in the corresponding tile 2D relationship file are connected.

Identifier: EJID

Code: 0022

Data Type: text numerals

Length: 20 characters

Format: unsigned integer64 as character string

Range: 0 to ( $2^{64} - 1$ )

Usage Note: Mandatory for all features belonging to Topological Network Datasets. Attribute is stored as a character string representing an unsigned 64-bit number, and requires conversion back into numerical representation by client reader. This is due to the 32-bit limitation on integer values within dBASE files.

Unit: N/A

Default: None

Compatibility: CDB 3.0

### 5.7.1.3.24 Feature Attribute Classification Code (FACC)

Description: This code used to distinguish and categorize features within a dataset. The enumerated codes are listed in /CDB/Metadata/Feature\_Data\_Dictionary.xml.

Identifier: FACC

Code: 0023

Data Type: text

Length: 5 characters

Format: two alpha characters following by three digits  
Range: N/A  
Usage Note: Mandatory  
Unit: N/A  
Default: None  
Compatibility: CDB 3.0, DIGEST 2.1

#### **5.7.1.3.25 FACC Sub Code (FSC)**

Description: This code, in conjunction with the FACC is used to distinguish and categorize features within a dataset. The enumerated codes are in accordance to Appendix N.

Identifier: FSC  
Code: 0024  
Data Type: numeric  
Length: 3 characters  
Format: integer. Enumerated per Appendix N  
Range: 0 to 999  
Usage Note: Mandatory  
Unit: N/A  
Default: None  
Compatibility: CDB 3.0

#### **5.7.1.3.26 Gate ID (GAID)**

Description: A unique alphanumeric identifier (for the airport in question) that is consistent with the IDENT attribute name within the NavData Gate dataset. This ID is the value of the Gate Identifier of the Gate dataset and can be used to extract additional information such as the gate position and bearing.

Identifier: GAID  
Code: 0025  
Data Type: alphanumeric  
Length: 6 characters  
Format: N/A  
Range: N/A



Usage Note:	Recommended and Optional usages are per Table 5-27: Allocation of CDB Attributes to Vector Datasets. Typically used (but not limited to) for models such as docking systems, marshallers and other models that are logically associated with a Terminal gate and that require some level of control by the simulation application.
Unit:	N/A
Default:	None
Compatibility:	CDB 3.1

### 5.7.1.3.27 Geomatics Extended Attribute Index (GEAI)

Description:	An index that points to a row entry of a Geomatics Extended Attribution file for the current dataset. This entry permits users to store an index to a link list set of Geomatics-specific attributes. CDB-compliant devices are not mandated to read and interpret this field. Usage of this attribution is not portable to other simulators because it falls outside of the documented CDB attribution scheme. The Geomatics Extended Attribution file should be located in the same directory as the instance-level attribution file. An empty GEAI attribute is allowed. Note that the first entry in the Geomatics Extended Attribution file has an index of 1.
Identifier:	GEAI
Code:	0026
Data Type:	numeric
Length:	6 characters
Format:	integer
Range:	1 to 999,999
Usage Note:	Optional. Use when Geomatics extended attribution is required. A “blank” or a value of 0 indicates that there are no Geomatics Extended attributes.
Unit:	N/A
Default:	None
Compatibility:	CDB 3.0.

### 5.7.1.3.28 Height above Surface Level (HGT)

Description:	The height of a feature. If the feature has no modeled representation, its height is measured as the distance from the surface level (ground
--------------	--

or water) to the tallest point of the feature above the surface<sup>66</sup>. If the feature has an associated 3D model, the height is measured as the distance from the XY plane of the model to the highest point of the model above that plane. HGT values are positive numbers.

Identifier:	HGT
Code:	0027
Data Type:	numeric
Length:	7 characters
Format:	floating-point (recommended precision 4.2)
Range:	0.00 to 9999.99
Usage Note:	In the case of ground features, HGT refers to the portion of the feature (or its modeled representation) that is meant to be above ground. In the case of network lineal and areal features such as roads, railroads, powerlines, or forest, HGT refers to the elevation of the feature relative to the terrain in its immediate vicinity.
Unit:	meters
Default:	0.00
Compatibility:	CDB 3.0

#### 5.7.1.3.29 Junction ID (JID)

**Description:** A Junction Identification Number that is used to virtually connect a point or an areal feature to another point, lineal or areal feature. Features stored in the same shape file having the same JID are connected. Features stored in different shape files having the same JID as the JID listed in the corresponding tile 2D relationship file are connected. When JID is associated to an areal feature, it necessarily connects to the first point of the areal feature.

Identifier:	JID
Code:	0028
Data Type:	text numerals
Length:	20 characters
Format:	unsigned integer64 as character string
Range:	0 to ( $2^{64} - 1$ )

---

<sup>66</sup> Surface here refers to the terrain in the immediate vicinity of the feature.



Usage Note: Mandatory for all features belonging to Topological Network Datasets. Attribute is used in 2D relationship file. Attribute is stored as a character string representing an unsigned 64-bit number, and requires conversion back into numerical representation by client reader. This is due to the 32-bit limitation on integer values within dBASE files.

Unit: N/A

Default: None

Compatibility: CDB 3.0

### 5.7.1.3.30 Location Accuracy (LACC)

Description: A precision value used to quantify the relative precision of the Location point representing the specific GeoPolitical Location.

Identifier: LACC

Code: 0029

Data Type: numeric

Length: 3 characters

Format: integer

Range: 0 to 999

Usage Note: Optional. See Table 5-20: Location Accuracy Enumeration Values for a list of accepted values.

Unit: meters.

Default: 0

Compatibility: CDB 3.0

**Table 5-20: Location Accuracy Enumeration Values**

LACC Code	Description
0	Unknown
1	Better or equal to 10 m.
2	Better or equal to 100 m.
3	Better or equal to 250 m.
4	Better or equal to 500 m.
5	Better or equal to 1200 m.
6	Greater than 1200 m.
997	Unpopulated
998	Not Applicable
999	Other

#### **5.7.1.3.31 Length of Lineal (LENL)**

**Description:** The length of a lineal. If the feature has been clipped to a tile boundary, the length still gives the initial full length of the object prior to the clipping operation, and if it belonged to a topological network, LENL will represent the distance between the two closest junction points encompassing this lineal segment. Note the Length attribute is not used to define a bounding sphere associated to an object, but rather to provide a weight to the relative length of the lineal as compared to others.

**Identifier:** LENL

**Code:** 0030

**Data Type:** numeric

**Length:** 6 characters

**Format:** integer

**Range:** 0 to 999,999

**Usage Note:** Mandatory for all networked lineal features. Length computation should account for the earth's curvature.

**Unit:** meters

**Default:** None

**Compatibility:** CDB 3.0, SEDRIS (EA = 562)

#### **5.7.1.3.32 Light Material Index (LMIX) – Deprecated**

**Description:** Index into the Composite Material Table that is used to determine the Light Material composition of the associated city illumination.



Represent the predominant material characterizing the major light attributes of a populated area.

Identifier:	LMIX
Code:	0031
Data Type:	numeric
Length:	6 characters
Format:	integer
Range:	0 to 999,999
Usage Note:	Optional. Applicable to Geopolitical Dataset areal features that delineate inhabited areas.
Unit:	N/A
Default:	None
Compatibility:	CDB 3.0

### 5.7.1.3.33 Feature (or Location) Name (LNAME)

**Description:** A toponym – a general term for any place or geographical entity. The attribute is used to give a proper noun (a human readable name) to any feature from any vector dataset.

Identifier:	LNAME
Code:	0032
Data Type:	text
Length:	32 characters
Format:	lexical
Range:	N/A
Usage Note:	The use of LNAME goes from the name of a City to the name of a Road, to the name of a Building, etc. Multiple names are possible when using LNAME as an extended attribute. When more than one name is provided, they must appear in order from the shortest name to the longest one.
Unit:	N/A
Default:	Blank
Compatibility:	CDB 3.0



#### 5.7.1.3.34 Location Type (LOTY)

Description:	A value that uniquely attributes a location feature according to the enumerators found here.
Identifier:	LOTY
Code:	0033
Data Type:	Enumeration per Table 5-21: Location Type Enumeration Values
Length:	3 characters
Format:	integer
Range:	0 to 999
Usage Note:	Optional. Applicable to Geopolitical Dataset areal features. See table below for a list of accepted values. Can be used only with Location Point, Lineal or Areal Feature Datasets (which are part of the Geopolitical Datasets)
Unit:	In cases where the location represents a bounded area, the approximate geometric center is assumed.
Default:	0
Compatibility:	CDB 3.0.

**Table 5-21: Location Type Enumeration Values**

LOTY Code	Description
0	Unknown
1	Continent
2	Country
3	State
4	Capital
5	Province
6	City
7	Municipality
997	Unpopulated
998	Not Applicable
999	Other

#### 5.7.1.3.35 Light Phase (LPH)

Description:	Used for all light types that are periodic in nature (rotating, blink, flashing, etc). The value of LPH controls the phase of the light relative to all other lights that share the same LTYP. All other light
--------------	--



characteristics, including frequency and duration are implicitly determined by the LTYP.

Identifier:	LPH
Code:	0034
Data Type:	numeric
Length:	3 characters
Format:	integer
Range:	0 to 999
Usage Note:	Optional. In absence of a value, LPH defaults to a value of 0.
Unit:	thousands of a cycle
Default:	000
Compatibility:	CDB 3.0

#### 5.7.1.3.36 Layer Priority Number (LPN)

Description:	A priority number that establishes the relative priority of overlapping features. LPN establishes the order (starting from 0 for lowest priority) by which overlapping features are processed by client-devices.
Identifier:	LPN
Code:	0035
Data Type:	numeric
Length:	5 characters
Format:	integer
Range:	0 to 32767
Usage Note:	Mandatory for terrain constraint features that overlap one another. For all other features, it is optional. LPN is derived from priority information stored and maintained by the authoring tools.
Unit:	N/A
Default:	None
Compatibility:	CDB 3.0

#### 5.7.1.3.37 Lane/Track Number (LTN)

Description:	The number of lanes on a road, tracks on railroad, or conductors on powerlines, including both directions.
--------------	--

Identifier:	LTN
Code:	0036
Data Type:	numeric
Length:	2 characters
Format:	integer
Range:	0 to 99 (Note: differs from DIGEST range of -32767 to 32768)
Usage Note:	Recommended for Road, RailRoad, and PowerLine Network features. Optional for Hydrography Network features.
Unit:	N/A
Default:	02 – for RoadNetwork lineal features 01 – for RailRoadNetwork lineal features 02 – for PowerLineNetwork lineal features
Compatibility:	CDB 3.0, DIGEST 2.1

#### **5.7.1.3.38 Light Type (LTYP)**

Description:	A unique code corresponding to a Light Type; Appendix E of this Specification provides the supported light types. The light types follow a hierarchical organization provided by the light type naming conventions described in Section 2.3, Light Naming. The Lights.xml file establishes the correspondence between the LTYP code and the Light Type name.
Identifier:	LTYP
Code:	0037
Data Type:	numeric
Length:	4 characters
Format:	integer
Range:	0 to 9999
Usage Note:	Mandatory for all Airport Light Point features, Environmental Light Point features.
Unit:	N/A
Default:	0
Compatibility:	CDB 3.0

#### **5.7.1.3.39 Model Level Of Detail (MLOD)**

Description:	The level of detail of the 3D model associated with the point feature.
--------------	--



Identifier:	MLOD
Code:	0038
Data Type:	numeric
Length:	3 characters
Format:	integer
Range:	-10 to 23
Usage Note:	When used in conjunction with MODL, the MLOD attribute indicates the LOD where the corresponding MODL is found. In this case, the value of MLOD can never be larger than the LOD of the Vector Tile-LOD that contains it. When used in the context of Airport and Environmental Light Point features, the value of MLOD, if present, indicates that this lightpoint also exist in a 3D model found at the specified LOD. In such case, the value of MLOD is not constrained and can indicate any LOD.
Unit:	N/A
Default:	None
Compatibility:	CDB 3.0

#### 5.7.1.3.40 Moving Model DIS Code (MMDC)

Description:	A character string composed of the 7 fields of the DIS Entity Type.
Identifier:	MMDC
Code:	0039
Data Type:	text
Length:	maximum of 29 characters
Format:	All seven fields of the DIS Entity Type separated by an underscore character (“_”): 1_2_3_4_5_6_7
Range:	N/A
Usage Note:	Mandatory for Moving Model Location features
Unit:	N/A
Default:	None
Compatibility:	CDB 3.0

#### 5.7.1.3.41 Model Name (MODL)

Description:	A string reference, the model name, which stands for the modeled geometry of a feature; in the case of buildings, this includes both its external shell and modeled interior.
Identifier:	MODL
Code:	0040
Data Type:	text
Length:	32 characters
Format:	per conventions described in Chapter 3, CDB Structure.
Range:	N/A
Usage Note:	Needed for Point features, Road Figure Point features, Railroad Figure Point features, Powerline Figure Point features and Hydrography Figure Point features that are modeled as OpenFlight or as RCS (Shape). MODL can also be used with Road Lineal features, Railroad Lineal features, Powerline Lineal features and Hydrography Lineal and Areal features. Note that it is not permitted to specify a value for MODL simultaneously with a value for MMDC.
Unit:	N/A
Default:	None
Compatibility:	CDB 3.0

#### 5.7.1.3.42 Model Type (MODT)

Description:	Indicates whether a feature is represented using a geotypical or geospecific model. Together, the MODT, FACC, FSC, and MODL attributes identify a unique model into the CDB.
Identifier:	MODT
Code:	0041
Data Type:	text
Length:	1 character
Format:	“T” for geotypical, “S” for geospecific
Range:	N/A
Usage Note:	Needed for features that are modeled as OpenFlight or as RCS (Shape).
Unit:	N/A
Default:	”S”



Compatibility: CDB 3.0

#### 5.7.1.3.43 Network Component Selector 1 (NCS1)

Description: Code that is used to identify the component selector 1 file which contain the point, lineal, or areal feature that is virtually connected.

Identifier: NCS1

Code: 0042

Data Type: numeric

Length: 4 characters

Format: unsigned integer

Range: 0 to 9999

Usage Note: Mandatory for Network datasets. Attribute is used in 2D relationship file.

Unit: N/A

Default: None

Compatibility: CDB 3.0

#### 5.7.1.3.44 Network Component Selector 2 (NCS2)

Description: Code that is used to identify the component selector 2 file which contain the point, lineal, or areal feature that is virtually connected.

Identifier: NCS2

Code: 0043

Data Type: numeric

Length: 4 characters

Format: unsigned integer

Range: 0 to 9999

Usage Note: Mandatory for Network datasets. Attribute is used in 2D relationship file.

Unit: N/A

Default: None

Compatibility: CDB 3.0

#### 5.7.1.3.45 Network Dataset Code (NDSC)

Description:	Code that is used to identify the dataset code file which contain the point, lineal, or areal feature that is virtually connected.
Identifier:	NDSC
Code:	0044
Data Type:	numeric
Length:	4 characters
Format:	unsigned integer
Range:	0 to 9999
Usage Note:	Mandatory for Network datasets. Attribute is used in 2D relationship file.
Unit:	N/A
Default:	None
Compatibility:	CDB 3.0

#### 5.7.1.3.46 Number of Instances (NIS) – Deprecated

Description:	Number of instances found in the corresponding 3D model associated with the cultural point feature.
Identifier:	NIS
Code:	0045
Data Type:	numeric
Length:	6 characters
Format:	integer
Range:	0 to 999,999
Usage Note:	Mandatory for features that are modeled as OpenFlight.
Unit:	N/A
Default:	None
Compatibility:	CDB 3.0

#### 5.7.1.3.47 Number of Indices (NIX) – Deprecated

Description:	Number of Indices – Number of indices found in the corresponding 3D model associated with the cultural point feature.
Identifier:	NIX
Code:	0046





Data Type:	numeric
Length:	6 characters
Format:	integer
Range:	0 to 999,999
Usage Note:	Mandatory for features that are modeled as OpenFlight.
Unit:	N/A
Default:	None
Compatibility:	CDB 3.0

### 5.7.1.3.48 Number of Normals (NNL) – Deprecated

Description:	Number of normal vectors found in the corresponding 3D model associated with the cultural point feature.
Identifier:	NNL
Code:	0047
Data Type:	numeric
Length:	6 characters
Format:	integer
Range:	0 to 999,999
Usage Note:	Mandatory for features that are modeled as OpenFlight.
Unit:	N/A
Default:	None
Compatibility:	CDB 3.0

### 5.7.1.3.49 Number of Texture Coordinates (NTC) – Deprecated

Description:	Number of Texture Coordinates – Number of texture coordinates found in the corresponding 3D model associated with the cultural point feature.
Identifier:	NTC
Code:	0048
Data Type:	numeric
Length:	6 characters
Format:	integer
Range:	0 to 999,999

Usage Note: Mandatory for features that are modeled as OpenFlight.  
Unit: N/A  
Default: None  
Compatibility: CDB 3.0

#### **5.7.1.3.50 Number of Texel (NTX) – Deprecated**

Description: Number of texels found in the corresponding 3D model associated with the cultural point feature.  
Identifier: NTX  
Code: 0049  
Data Type: numeric  
Length: 9 characters  
Format: integer  
Range: 0 to 999,999,999  
Usage Note: Mandatory for features that are modeled as OpenFlight.  
Unit: N/A  
Default: None  
Compatibility: CDB 3.0

#### **5.7.1.3.51 Number of Vertices (NVT)**

Description: Number of Vertices – Number of vertices of the corresponding 3D model associated with the cultural point feature.  
Identifier: NVT  
Code: 0050  
Data Type: numeric  
Length: 6 characters  
Format: integer  
Range: 0 to 999,999  
Usage Note: This attribute depends on the presence of a model (MODL); a good approximation is the number of vertices in the vertex pool of the model.  
Unit: N/A  
Default: None  
Compatibility: CDB 3.0

**5.7.1.3.52 Population Density (POPD)**

Description:	The number of inhabitants per square kilometer.
Identifier:	POPD
Code:	0051
Data Type:	numeric
Length:	5 characters
Format:	integer
Range:	0 to 99999
Usage Note:	Applicable to Geopolitical features representing inhabited areas.
Unit:	Inhabitants per square kilometer
Default:	0
Compatibility:	CDB 3.0

**5.7.1.3.53 Populated Place Type (POPT)**

Description:	A value that uniquely represents the Populated Place Attribution Type. This attribute should be used in conjunction with the BOTY attribute when BOTY has an (enumerator) value of 13 which corresponds to “Populated Place”
Identifier:	POPT
Code:	0052
Data Type:	Enumeration per Table 5-22: Populated Place Type Enumeration Values
Length:	3 characters
Format:	integer
Range:	0 to 999
Usage Note:	Optional. Applies to Geopolitical Dataset areal features that delineate inhabited areas. See table below for a list of accepted values.
Unit:	N/A
Default:	0
Compatibility:	CDB 3.0

**Table 5-22: Populated Place Type Enumeration Values**

<b>POPT Code</b>	<b>Description</b>
0	Unknown
1	Native Settlement
2	Shanty Town
3	Tent Dwellings
4	Inland Village
5	Small City (less than 20,000 inhabitants)
6	Medium City (between 20,000 and 500,000 inhabitants)
7	Large City (more than 500,000 inhabitants)
997	Unpopulated
998	Not Applicable
999	Other

#### **5.7.1.3.54 Relative Tactical Importance (RTAI)**

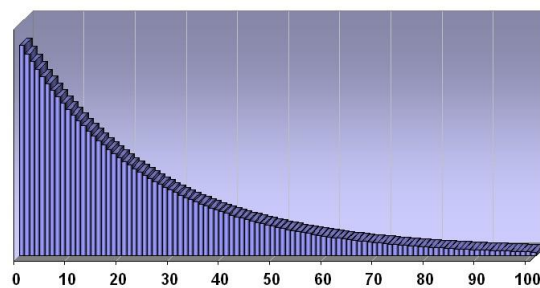
**Description:** Provides the Relative TActical Importance of cultural features relative to other features for the purpose of client-device scene/load management<sup>67</sup>. A value of 100% corresponds to the highest importance; a value of 0% corresponds to the lowest importance. When confronted with otherwise identical objects that differ only wrt to their Relative TActical Importance, client-devices should always discard features with lower importance before those of higher importance in the course of performing their scene / load management function. As a result, a value of zero gives complete freedom to client-devices to discard the feature as soon as the load of the client-device is exceeded. The effectiveness of scene / load management functions can be severely hampered if large quantities of features are assigned the same Relative TActical Importance by the modeler. In effect, if all features are assigned the same value, the client-devices have no means to distinguish tactically important objects from each other. Assigning a value of 1% to all objects is equivalent to assigning them all a value of 99%. Ideally, the assignment of tactical importance to features should be in accordance to a histogram similar to the one shown here. The shape of the curve is not critical, however the proportion of features tagged with a high importance compared to those with low importance is critical in achieving effective scene/load management schemes. It is illustrated here to show that few features should have an importance

<sup>67</sup> Note that the importance of the model can be further modified at run-time at the simulator console through the scenario importance value assigned to the model.

of 100 with progressively more features with lower importance. The assignment of the RTAI to each feature lends itself to database tools automation. For instance, RTAI could be based on a look-up function which factors the feature's type (FACC or MMDC).

$$RTAI = f(FACC \text{ _ or _ } MMDC)$$

The value of Relative Tactical Importance should be accounted for by client-devices (in combination with other information) to determine the appropriate distance at which the feature (and its modeled representation, if available) should be rendered or processed. Relative Tactical Importance is mandatory. It has no default value.



**Figure 5-31: RTAI Typical Usage Histogram**

Identifier:	RTAI
Code:	0053
Data Type:	numeric
Length:	3 characters
Format:	integer
Range:	0 to 100
Usage Note:	Mandatory. All features should be tagged with an appropriate value for the reasons stated above.
Unit:	Percentage
Default:	None
Compatibility:	CDB 3.0

#### **5.7.1.3.55 Runway ID (RWID)**

Description:	An alphanumeric identifier that uniquely identifies a runway for a given airport; this ID must match the value of the field Ident of the Runway or Helipad dataset.
Identifier:	RWID

Code:	0054
Data Type:	Alphanumeric
Length:	6 characters
Format:	N/A
Range:	N/A
Usage Note:	Recommended for all Airport Light Points features. Failure to appropriately tag airport culture with RWID attribute will result in reduced control of runway-related (or helipad) culture by simulator. Optional for Point/Lineal/Areal features, Location Points Features, Environmental Light Point features, and Moving Model Location features that are associated with a runway and for which control of the feature is desirable. The combination of RWID and APID points to a unique record of the NavData Runway or Helipad dataset components. Note that all of the lights and other features located in vector datasets that are associated with the operation of a runway or helipad are required to reference a runway or helipad in the NavData dataset; the RWID attribute is not required for features unrelated to a runway or helipad.
Unit:	N/A
Default:	None
Compatibility:	CDB 3.0

#### 5.7.1.3.56 Scaling (SCALx)

#### 5.7.1.3.57 Scaling (SCALy)

#### 5.7.1.3.58 Scaling (SCALz)

Description:	A set of scaling factors, one of the model axis, to be applied to the rendering of model geometry by the client-device. A value of 1.0 instructs the client-devices to use the model as-is. The physical dimension of models processed by client-device should approach zero, as SCALing tends to zero. The value of SCALing should also be accounted for by client-devices (in combination with other information) to determine the appropriate distance at which the model should be paged-in, rendered or processed. All three SCALing factors are optional. Values of zero and negative values are not permitted.
Identifiers:	SCLAx, SCALy, SCALz
Codes:	0055, 0056, 0057



Data Type:	numeric
Length:	9 characters
Format:	floating-point (recommended precision 3.5)
Range:	000.00001 to 999.99999
Usage Note:	Optional. CDB readers should default to a value of 1.0 if SCALx, SCALy, or SCALz is missing.
Unit:	N/A
Default:	1.0
Compatibility:	CDB 3.0

### 5.7.1.3.59 Start Junction ID (SJID)

Description:	A Junction Identification Number that is used to virtually connect the start point of a lineal to another point, lineal or areal feature. Lineal features stored in the same shape file having the same SJID or EJID are connected. Lineal features stored in different shape files having the same SJID or EJID as the JID listed in the corresponding tile 2D relationship file are connected.
Identifier:	SJID
Code:	0058
Data Type:	text numerals
Length:	20 characters
Format:	unsigned integer64 as character string
Range:	0 to $(2^{64} - 1)$
Usage Note:	Mandatory for all features belonging to Topological Network Datasets. Attribute is stored as a character string representing an unsigned 64-bit number, and requires conversion back into numerical representation by client reader. This is due to the 32-bit limitation on integer values within dBASE files.
Unit:	N/A
Default:	None
Compatibility:	CDB 3.0

### 5.7.1.3.60 Surface Roughness Description (SRD)

Description:	Describes the condition of the surface materials that may be used for mobility prediction, construction material, and landing sites.
Identifier:	SRD



Code: 0059

Data Type: Enumeration per Table 5-23: Surface Roughness Enumeration Values

Length: 3 characters

Format: integer

Range: 0 to 999

Usage Note: Recommended for Areal features.

Unit: N/A

Default: 0

Compatibility: CDB 3.0, DIGEST 2.1

**Table 5-23: Surface Roughness Enumeration Values**

<b>SRD Code</b>	<b>Description</b>
0	Unknown
1	No surface roughness effect
2	Area of high landslide potential
3	Uncohesive surface material/flat
4	Rough
5	Angular
6	Rounded
11	Surface of numerous cobbles and boulders
12	Areas of stony terrain
13	Stony soil with surface rock
14	Stony soil with scattered boulders
15	Stony soil with numerous boulders
16	Numerous boulders
17	Numerous rock outcrops
18	Area of scattered boulders
19	Talus slope
20	Boulder Field
31	Highly fractured rock surface
32	Weathered lava flows
33	Unweathered lava flows
34	Stony soil with numerous rock outcrops
35	Irregular surface with deep fractures of foliation
36	Rugged terrain with numerous rock outcrops



<b>SRD Code</b>	<b>Description</b>
37	Rugged bedrock surface
38	Sand dunes
39	Sand dunes/low
40	Sand dunes/high
41	Active sand dunes
42	Stabilized sand dunes
43	Highly distorted area, sharp rocky ridges
51	Stony soil cut by numerous gullies
52	Moderately dissected terrain
53	Moderately dissected terrain with scattered rock outcrops
54	Dissected floodplain
55	Highly dissected terrain
56	Area with deep erosional gullies
57	Steep, rugged, dissected terrain with narrow gullies
58	Karst areas of numerous sinkholes and solution valleys
59	Karst area of numerous sinkholes
60	Karst/hummocky terrain covered with large conical hills
61	Karst/hummocky terrain covered with low, broad-based mounds
62	Arroyo/wadi/wash
63	Playa/dry lake
64	Area of numerous meander scars and/or oxbow lakes
65	Solifluction lobes and frost scars
66	Hummocky ground, areas of frost heaving
67	Area of frost polygons
68	Area containing sabkhas
69	Area of numerous small lakes and ponds
70	Area of numerous crevasses
81	Area of numerous terraces
82	Quarries
83	Strip mines
84	Quarry/gravel pit
85	Quarry/sand pit
86	Mine tailings/waste piles
87	Salt evaporators
88	Area of numerous dikes
89	Area of numerous diked fields
90	Area of numerous fences

SRD Code	Description
91	Area of numerous stone walls
92	Area of numerous man-made canals/drains/ditches
93	Area of numerous terraced fields
94	Parallel earthen mounds row crops
95	Area of numerous hedgerows
997	Unpopulated
998	Not Applicable
999	Other

#### 5.7.1.3.61 Structure Shape Category (SSC)

Description:	Describes the Geometric form, appearance, or configuration of the feature.
Identifier:	SSC
Code:	0060
Data Type:	Enumeration per Table 5-24: Structure Shape Category Enumeration Values
Length:	3 characters
Format:	integer
Range:	0 to 999
Usage Note:	Recommended for Point features, and all Network Lineal/Areal Figure Points features.
Unit:	N/A
Default:	0
Compatibility:	CDB 3.0, DIGEST 2.1

**Table 5-24: Structure Shape Category Enumeration Values**

SSC Code	Description
0	Unknown
1	Barrel, Ton
2	Blimp
3	Boat Hull (Float)
4	Bullet
5	<i>Reserved</i>
6	Conical/Peaked/NUN



SSC Code	Description
7	Cylindrical (Upright)/CAN
9	<i>Reserved</i>
10	Pillar/Spindle
11	<i>Reserved</i>
12	Pyramid
13	<i>Reserved</i>
14	<i>Reserved</i>
15	Solid/filled
16	Spar
17	Spherical (Hemispherical)
18	Truss
19	With Radome
20	<i>Reserved</i>
21	Artificial Mountain
22	Crescent
23	Ferris Wheel
24	Enclosed
25	Roller Coaster
26	Lateral
27	Mounds
28	Ripple
29	Star
30	Transverse
31	<i>Reserved</i>
32	<i>Reserved</i>
33	<i>Reserved</i>
34	<i>Reserved</i>
36	Windmotor
38	<i>Reserved</i>
40	<i>Reserved</i>
46	Open
52	'A' Frame
53	'H' Frame
54	'T' Frame
56	'Y' Frame
57	<i>Reserved</i>
58	<i>Reserved</i>

SSC Code	Description
59	Telescoping Gasholder (Gasometer)
60	Mast
61	Tripod
62	<i>Reserved</i>
63	<i>Reserved</i>
65	Cylindrical with flat top
66	Cylindrical with domed top
71	Cylindrical/Peaked
73	Superbuoy
74	'T' Frame
75	Tetrahedron
76	Funnel
77	Arch
78	Multi-Arch
79	Round
80	Rectangular
81	Dragons Teeth
82	I-Beam
83	Square
84	Irregular
85	Diamond Shaped Buoy
86	Oval
87	Dome
88	Spherical with Column Support
89	Cylindrical or Peaked with tower support
90	High-Rise Building
91	Cylindrical
92	Cubic
93	Pole
94	Board
95	Column (Pillar)
96	Plaque
97	Statue
98	Cross
107	Tower
108	Scanner
109	Obelisk



SSC Code	Description
110	Radome, Tower Mounted
997	Unpopulated
998	Not Applicable
999	Other

#### 5.7.1.3.62 Structure Shape of Roof (SSR)

Description:	Describes the roof shape.
Identifier:	SSR
Code:	0061
Data Type:	Enumeration per Table 5-25: Structure Shape of Roof Enumeration Values
Length:	3 characters
Format:	integer
Range:	0 to 999
Usage Note:	Recommended for Point features, and all Network Lineal/Areal Point Figures.
Unit:	N/A
Default:	0
Compatibility:	CDB 3.0, DIGEST 2.1

**Table 5-25: Structure Shape of Roof Enumeration Values**

SSR Code	Description
0	Unknown
6	Conical/Peaked/NUN
38	Curved/Round (Quonset)
40	Dome
41	Flat
42	Gable (Pitched)
43	<i>Reserved</i>
44	<i>Reserved</i>
45	<i>Reserved</i>
46	<i>Reserved</i>
47	Sawtooth
48	<i>Reserved</i>

SSR Code	Description
49	<i>Reserved</i>
50	With Monitor
51	With Steeple
55	Flat with Monitor
58	<i>Reserved</i>
65	Gable with Monitor
65	<i>Reserved</i>
66	<i>Reserved</i>
71	<i>Reserved</i>
72	<i>Reserved</i>
77	With Cupola
78	With Turret
79	With Tower
80	With Minaret
997	Unpopulated
998	Not Applicable
999	Other

#### 5.7.1.3.63 Traffic Flow (TRF)

Description: Encodes the general destination of traffic.

Identifier: TRF

Code: 0062

Data Type: numeric

Length: 3 characters

Format: Integer. Enumerated per DIGEST 2.1. A few examples:  
3: One-way  
4: Two-way

Range: 0 to 999

Usage Note: Recommended on all Network Lineal (except PowerLines) features.

Unit: N/A

Default: 003 – for RailRoadNetwork lineal features  
004 – for other network lineal features

Compatibility: CDB 3.0, DIGEST 2.1



**5.7.1.3.64 Taxiway ID (TXID)**

Description:	A unique alphanumeric identifier (for the airport in question).
Identifier:	TXID
Code:	0063
Data Type:	alphanumeric
Length:	6 characters
Format:	N/A
Range:	N/A
Usage Note:	Recommended usage and Optional usages are per table Table 5-27: Allocation of CDB Attributes to Vector Datasets. Failure to appropriately tag airport culture with TXID attribute will result in reduced control of taxiway-related culture by a simulation device.
Unit:	N/A
Default:	None
Compatibility:	CDB 3.1

**5.7.1.3.65 Urban Street Pattern (USP)**

Description:	Describes the predominant geometric configuration of streets found within the delineated area of the feature.
Identifier:	USP
Code:	0064
Data Type:	Enumeration per Table 5-26: Urban Street Pattern Enumeration Values
Length:	3 characters
Format:	integer
Range:	0 to 999
Usage Note:	Recommended for Areal features.
Unit:	N/A
Default:	0
Compatibility:	CDB 3.0, DIGEST 2.1

**Table 5-26: Urban Street Pattern Enumeration Values**

USP Code	Description
0	Unknown

USP Code	Description
2	Rectangular/Grid-Regular
3	Rectangular/Grid-Irregular
4	Curvilinear (cluster)
6	Concentric / Radial-Regular
7	Concentric / Radial-Irregular
9	Mixed-Curvilinear (cluster) and Rectangular (grid)
10	Mixed-Concentric / Radial and Rectangular (grid)
11	Mixed-Curvilinear (cluster) and Concentric / Radial
12	<i>Reserved</i>
13	Linear Strip
997	Unpopulated
998	Not Applicable
999	Other

#### 5.7.1.3.66 Vendor Extended Attribute Index (VEAI)

**Description:** An index that points to a row entry of a VendorExtended Attribution file for the current dataset. This entry permits users to store an index to a link list set of Vendor-specific attributes. CDB-compliant devices are not mandated to read and interpret this field. Usage of this attribution is not portable to other simulators because it falls outside of the documented CDB attribution scheme. The Vendor Extended Attribution file should be located in the same directory as the instance-level attribution file. An empty VEAU attribute is allowed. Note that the first entry in the Vendor Extended Attribution file has an index of 1.

**Identifier:** VEAU

**Code:** 0065

**Data Type:** numeric

**Length:** 6 characters

**Format:** integer

**Range:** 1 to 999,999

**Usage Note:** Optional. Use when Vendor extended attribution is required. A “blank” or a value of 0 indicates that there are no Vendor Extended attributes.

**Unit:** N/A

**Default:** None

Compatibility: CDB 3.0.

#### **5.7.1.3.67 Width with Greater Than 1 meter Precision (WGP)**

**Description:** For lineal features (such as roads, railways, runways, taxiways), WGP is a measurement of the shorter of two linear axes. For a bridge, the width is the measurement perpendicular to the axis between the abutments. For powerlines, the width is the distance between the outermost wires.

**Identifier:** WGP

**Code:** 0066

**Data Type:** numeric

**Length:** 9 characters

**Format:** floating-point (recommended precision 5.3)

**Range:** 0.000 to 99,999.999

**Usage Note:** Recommended on all Network Lineal features.

**Unit:** meters

**Default:** None

**Compatibility:** CDB 3.0, DIGEST 2.1

#### **5.7.1.4 Explicitly Modeled Representations**

##### **5.7.1.4.1 Referenced by Point Features**

A point feature (whose position and attributes are stored in a Shapefile) can also refer to an explicitly modeled representation.

A feature can point to an explicitly modeled representation of that feature that is stored in either the GTModel library, the MModel library or alternately embedded inside a CDB tile. In order to specify the modeled representation, the modeler must properly attribute the feature via the MODL, MLOD, MMDC and MODT attributes in the vector dataset that contains the feature. For Point features, the CDB supports two types of explicitly modeled representations:

- OpenFlight models
- RCS Shapefile models

Natural vector features (such as trees, bushes) are usually represented by geotypical OpenFlight models. The majority of man-made features can also be geotypical in nature. For instance, power pylons, telephone poles, or residential houses can all be represented with generic models that are typical in appearance to the real-world objects they represent. The modeler need only resort to a geospecific model if the

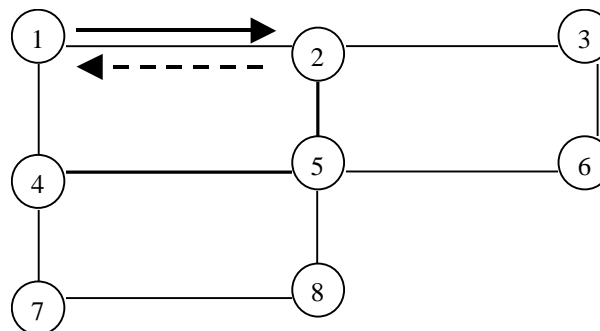
application requires a model with the unique shape, appearance and/or properties of the real-world object.

### 5.7.1.5 Implicitly Modeled Representations

An implicitly modeled representation is one that is defined completely by the supplied attribution of the Dataset in which it is contained. Examples of implicitly modeled features are light-points.

### 5.7.1.6 Handling of Topological Networks

The CDB provides several interconnected topological network datasets consisting of multiple datasets. Each network dataset can be made of separate point features and or a series of points connected together using lineal and areal features. Each lineal feature has a start and end nodes, which correspond to intersections when connected to two or more other lineal features, or connections when connected to an areal; the other intermediate points are used to accommodate deviation from a straight line. Typically, network datasets (such as roads, railroads) conform to the ground; consequently, when the optional AHGT attribute is present, its value is set to false. Each network dataset is stored as a distinct Shapefile.



**Figure 5-32: Example of a Topological Network**

The CDB Topological networks are useful when one needs to determine the shortest path between two arbitrary nodes in the entire network; alternately, algorithms can use the network topology in combination with a “cost” parameter based on length (in the case of shortest path), traffic speed (in the case of fastest path), or some other criteria that can be derived from the attribution information associated with the network datasets.

The CDB Topological networks are used for the following purposes:

- To determine a route for features such as roads, rivers, railroads.
- To follow a route made of roads, rivers, railroads.
- To avoid an obstacle; for example a tank may not be able to cross a river or be able to go over or under a pipeline.

- To efficiently process a “feature” for devices (such as radar) that do not require a high definition of the geometrical representation or do not need to represent more than one dimension.
- To represent features in a map view, where features have implicit classes and priority.

The CDB Topological networks are optimized to facilitate road/river/railroad following tasks. They support the notion of directionality such as one-way roads or both ways for two-way roads, rivers. The vertex positions are physically positioned along the center of the feature’s longitudinal axis; for example, a road such as a dual-lane undivided highway, the vertex data lies along the stripes dividing the two lanes.

Features within the same or different network datasets are connected together using the junction identifier attributes SJID, EJID or JID. Two or more features having the same identifier values are considered virtually connected. This junction identifier allows, for instance, a primary road to connect to a secondary road, or a river to connect a lake (in same network datasets), or to connect a road and a river (in different network dataset). SJID, EJID or JID are mandatory attributes for all topological network datasets. When not specified (i.e., blank), the feature is not connected to any other features. Appendix A provides guidelines on how to generate the junction identifiers.

Since the junction identifier is associated with a shape type feature, the following combinations are supported:

- Any point feature can be connected to any start or end point of a lineal feature (point to lineal connection), or to any start point of an areal feature (point to areal connection), using its JID attribute.
- Any start point of a lineal feature can be connected to any point feature (point to lineal connection), or to any start or end point of a lineal feature (lineal to lineal connection), or to any start point of an areal feature (lineal to areal connection), using its SJID attribute.
- Any end point of a lineal feature can be connected to any point feature (point to lineal connection), or to any start or end point of a lineal feature (lineal to lineal connection), or to any start point of an areal feature (lineal to areal connection), using its EJID attribute.
- Any start point of an areal feature can be connected to any point feature (point to areal connection), or to any start or end point of a lineal feature (lineal to areal connection), using its JID attribute.

Connection information between two features located in two separate Shapefiles is explicitly listed in 2D relationship files. This Specification currently supports two types of 2D relationship files; the 2D relationship tile connection file which specifies connections of the same dataset feature between two adjacent tiles, and the 2D relationship dataset connection file which specifies connection of 2 or more features from different dataset components within the same tile.

#### **5.7.1.6.1 2D Relationship Tile Connection File**

The CDB Topological network is broken into tiles and therefore must be clipped against tile boundaries. To ensure the connectivity between tile boundaries of a lineal feature, the resulting clipping point must share the same junction identifier (JID) in both tiles. This clipping point potentially exists in several Tile-LODs having a common boundary; in which case, all points representing the same clipping point share the same JID. Doing so ensures that connectivity between geocells and tiles is preserved. A clipping point can be identified by the application by checking the 2D relationship tile connection file. There is a 2D relationship tile connection file per network dataset tile. When the file is missing, it indicates there is no clipping point for the lineals belonging to the tile. The 2D relationship file is a dBASE file that contains a list of records made of 2 attributes; the Junction ID (JID) that identifies the start or end point of the clipped lineal and the Network Component Selector 1 (NCS1) that identifies the network dataset lineal file. The dataset code file is implicit to the network dataset tile directory and the Network Component Selector 2 always represents a lineal feature Shapefile (code 003) thus do not require to be included in the record. The coordinate of the tile adjacent to a clipping point can be determined using the latitude and longitude of that point.

If a connection between two lineal features happens to be located exactly at a tile boundary, the lineal is obviously not clipped but a junction ID is allocated and included in the 2D relationship tile connection file.

In a 2D relationship tile connection file, no two records are identical; however JIDs may appear more than once with different NCS1, indicating a connection between network dataset components.

#### **5.7.1.6.2 2D Relationship Dataset Connection File**

The CDB Topological network is made of several network datasets that in turn are made of several Shapefiles. By specifying a junction identifier per feature, any features in any of these several Shapefiles can in theory be connected to any other features located in a separate Shapefile. A connection within a tile, which includes the tile boundaries, can be identified by the application by checking the 2D relationship dataset connection file. There is a 2D relationship dataset connection file per network dataset tile. This file contains all the connections between the components of the corresponding network dataset with other network datasets. When the file is missing, it indicates there are no connections within the tile. The 2D relationship file is a dBASE file that contains a list of records made of 4 attributes; the Junction ID (JID) that identifies the connected point, lineal or areal features, the Network Dataset Code (NDSC) that identifies the network dataset the feature belongs to, and the Network Component Selectors (NCS1 and NCS2) that identify the network dataset component and the shape type. The tile coordinate and tile LOD is implicit to the Network Dataset tile directory.

All the records in the 2D relationship file are sorted per ascending JID. This has two advantages; it speeds up the search process when looking for a specific JID and it

groups all the features that are connected together. In effect, there is always a minimum of two consecutive records with the same JID; the record belonging to the corresponding file dataset (or component) and the record identifying the feature it connects to. Note that when a 2D relationship file specifies a connection to a feature belonging to different network dataset, the corresponding LOD file of this dataset may or may not exist. If the corresponding LOD file of the target dataset is missing, the application must look for the feature in the coarser LOD file of the target dataset. If the corresponding LOD file of the target dataset exists, the feature may be missing because it has been removed by the off-line tool decimation process; when this is the case, the application must look for the feature in the finer LOD file of the target dataset.

#### **5.7.1.6.3 Junction Identifier (SJID, EJID, and JID) Range**

This version of the Specification imposes that SJID, EJID and JID have unique values for all network datasets within the same geocell. With a 64-bit range, it is practically impossible to run out of ID number. In the process of creating the unique identifier, special care must be taken by the off line tool in order to avoid duplicating the identifier at the geocell boundary for the clipping points when the modified or added features overlap two or more geocells.

Table 3-27, CDB LOD versus Feature Density, specifies the maximum number of elements in a tile for vector datasets. This maximum number is not affected by the number of added clipping point in a lineal feature. Although, this appears to lead to an unbounded number of points in a file, it is clamped to the geocell size. In practice, for a relative constant density, the number of clipping points diminishes as the LOD number increases.

#### **5.7.1.6.4 Network Vector Priority**

When generating CDB Tile-LODs for lineal networks, there is also the concept of vector priority. This concept is to accommodate efficient path planning and following, as well as map drawing and other non-visual usages of networked lineals. The assurances implied by this vector priority concept are the following:

- The finest Tile-LOD for a lineal network contains all the features and geometry of that dataset.
- Coarser Tile-LODs contain both simplified lineal features as well as fewer features, such that:
  - There is a minimum priority class that exists within the Tile-LOD. Not all the features with this priority class may exist in this Tile-LOD.
  - All features in priority classes greater than the minimum must exist, but may be simplified from their full resolution version in the finest Tile-LOD. All topological connections between higher priority classes also exist in this Tile-LOD.
  - All features in priority classes less than the minimum do not exist in this Tile-LOD, and can be found in finer Tile-LODs.



- The maximum number of points for each Tile-LOD conforms to Table 3-27.

The default vector priority values can be found in the Feature Data Dictionary that accompanies the CDB Specification. They cover the Road, Railroad, Powerline, and Hydrography Network datasets. If there are two or more coincident lineal features, use the higher of the Vector Priority value on each feature. For example, if a bridge and a road lineal feature are coincident, use the higher vector priority value when creating the Tile-LODs so that either both exist at the same time or neither exists in the Tile-LOD. Appendix A contains the recommended way to create the Tile-LODs for lineal network datasets.

Areal network datasets are not covered by vector priority. They should be simplified into Tile-LODs using each feature's significant size and applying spatial significance criteria to each vertex.

#### **5.7.1.7 Handling of Light Points**

All of the information that is needed to instantiate the light point features is organized in accordance to the CDB terrain tile structure. Each instance of a light point feature refers to a light type defined by the CDB Specification via its shape attribution (LTYP). As a result, the entire definition of the light (i.e., its location, orientation and attributions) is self-contained within the shape files.

The CDB Specification defines a collection of CDB Light Types that includes airport/runway lighting systems, cultural lights, aircraft refueling systems, etc. The light types currently supported by the CDB Specification are listed in Appendix E of this Specification; they are also listed in Lights.xml as specified in Section 2.3, Light Naming. While the CDB Specification provides a rigorous definition for each type of light, its representation is entirely determined by the fidelity and the capabilities of client-devices.

#### **5.7.1.8 Allocation of CDB Attributes To Vector Datasets**

The CDB Specification limits the applicability of each of the CDB attributes to certain vector datasets. This approach helps to reduce the number of columns (hence to reduce the size) of the dBASE instance and class-level attribution files.

The allocation of CDB attributes to each of the Vector datasets is prescribed by Table 5-27 below.

### Table 5-27: Allocation of CDB Attributes to Vector Datasets

[illegible]

## GSFeature Dataset


Man Made Point	O	R	O	O	O	O	D	O	M	M	O	O						M	M	O	O	D			O					D	O				D		M	O	O	O		R	R	O			O
Man Made Lineal	O		O					O	M	M	O	O	R					M	M	O	O	R			O		M									M	O						O			O	R
Man Made Areal	O		O					O	M	M	O	O			R	R		M	M	O	O	R			O		M							M	O				O			O		R	O		
Natural Point	O	R	O	O	O	O	D	O	M	M	O	O						M	M	O	O	D			O				D	O			D	M	O	O	O							O			
Natural Lineal	O		O					O	M	M	O	O	R					M	M	O	O	R			O		M							M											O	R	
Natural Areal	O		O					O	M	M	O	O				R		M	M	O	O	R			O		M						M						R						O		
Tree Point	O	R	O	O	O	O	D	O	M	M	O	O						M	M	O	D			O				D	O			D	M	O	O	O							O				
Airport Light Point	O	R	R					O	M	M	O							M	M	R	O	O			O	O		M	O					M	R						R			O			
Airport Lineal	O		R					O	M	M	O	O	R					M	M	O	O	R			O		M						M	O							O			O	R		
Airport Areal	O		R					O	M	M	O	O			R	R		M	M	O	O	R			O		M						M	O				O						O			
Environmental Light Point	O	R	O					O	M	M	O							M	M	O	O	O			O	O		M	O				M	O							O			O			

## GTFeature Dataset

Man Made Point	O	R	O	O	O	O	D	O	M	M	O	O						M	M	O	O	D			O					D	O				D		M	O	O	O		R	R	O			O
Man Made Lineal	O		O					O	M	M	O	O	R					M	M	O	O	R			O		M								M	O						O			O	R	
Man Made Areal	O		O					O	M	M	O	O			R	R		M	M	O	O	R			O		M							M	O					O			O		R	O	
Tree Point	O	R	O	O	O	O	D	O	M	M	O	O						M	M	O	O	D			O				D	O			D		M	O	O	O					O				
Tree Lineal	O		O					O	M	M	O	O	R					M	M	O	R			O		M								M										O	R		
Tree Areal	O		O					O	M	M	O	O				R		M	M	O	O	R			O		M							M										O			
Moving Model Location	O	M	O	O	O	O		M	O	M	M	O	O						O	O	M			O					M				M				M	O					O		O		

## GeoPolitical Dataset



[illegible]

M	Mandatory	R	Recommended	D	Deprecated Attribution Schema	P	Preferred Attribution Schema		Not permitted
O	Optional	D	Dependent on other attribute	Y	Allowed Attribution Schema				Not used

(Cont'd on next page)

**Table 5-27: Allocation of CDB Attributes to Vector Datasets (Cont'd)**

[illegible]

<b>M</b>	Mandatory	<b>R</b>	Recommended	<b>D</b>	Deprecated Attribution Schema	<b>P</b>	Preferred Attribution Schema		Not permitted
<b>O</b>	Optional	<b>D</b>	Dependent on other attribute	<b>Y</b>	Allowed Attribution Schema				Not used



### **5.7.1.9 Vector Significant Size and Spatial Significance Criteria**

All vector features have an implicit or explicit significant size, which must be used when creating coarser Tile-LODs. In addition, as Tile-LODs are created, individual vertices within the vector also have a spatial significance within the feature itself.

#### **5.7.1.9.1 Vector Significant Size**

The significant size for point features is defined by the significant size of the feature or model it represents, as specified in Section 6.8.3. Lineal feature significant size is equivalent to the width of the lineal feature as defined by its WGP attribute.

Areal feature significant size is proportional to the diagonal of the feature's bounding box. If the feature is a box of equal length and width, its significant size is exactly the bounding box diagonal. As the shape of the feature's bounding box becomes more rectangular, and as the amount of negative space within the bounding box increases, the feature's significant size should be proportionally decreased relative to the departure from an equal length sided bounding box. This definition is similar to the one of a 3D model as specified in Section 6.8.3.2.

#### **5.7.1.9.2 Levels of Detail and Spatial Significance Criteria**

As coarser Tile-LODs of the lineal and areal datasets are created, the individual vertices of lineal and areal features must conform to the vector spatial significance criteria. Vertices must be moved or removed during coarser Tile-LOD creation such that, no part of the feature can be defined or changed by more than  $\frac{1}{2}$  of a raster cell size, as indicated by column 4 (Approximate Grid Spacing) of Table 2-4. For example, when creating the Tile-LOD 1 of a network dataset, all parts of each feature must be within 27 meters of the original feature (54.355 meters per Tile-LOD1 grid / 2 = 27.1775 meters). The spatial significance criterion is relaxed for the finest Tile-LOD, which is only limited by the feature vertex count.

### **5.7.2 Tiled Navigation Dataset**

As described in section 5.2, the global navigation dataset is complemented by a tiled-based dataset of basic navigation information that refers to the corresponding geocell. Those are found in the \401\_Navigation dataset which is subdivided into several components as listed in the next table.

**Table 5-28: Tiled Navigation Dataset**

CS1	CS2	File Extension	Component Name	Component Description
Dataset 401, Navigation				
001-046	001	*.shp *.shx *.dbf *.dbt	Tiled Navigation Dataset	Contains the basic Navigation records

Refer to Table 5-3: List of Navigation Components, for a complete description of the possible values of CS1.

#### **5.7.2.1 Default Read Value**

Simulator client-devices should assume an empty tile when data is not available.

#### **5.7.2.2 Default Write Value**

The files associated with this dataset for the area covered by the geocell need not be created if the source data is not available.

#### **5.7.3 Tiled GSFeature Dataset**

A GSFeature is a geospecific (point, lineal, or areal) feature whose optional modeled representation is also geospecific.

The GSFeature Dataset provides the position, size, orientation (points), shape (lineal and areals), type and attribution of point, lineal, and areal features. It is subdivided into the following components:

- CS1 = 001: Man-made features
- CS1 = 002: Natural features
- CS1 = 003: Trees features
- CS1 = 004: Airport features
- CS1 = 005: Environmental lights

Table 5-29: Component Selectors for GSFeature Dataset lists all of the components of the dataset. The allocation of CDB attributes to each of the components is prescribed by Table 5-27: Allocation of CDB Attributes to Vector Datasets.

It is customary in many simulation applications to represent street lighting as points of lights; as a result, Airport and Environmental light points can be entirely described by their feature position and attribution information and thus, do not have additional “modeled” data.

The modeling of geospecific trees is permitted when required to represent a specific geographic area; however, it is understood that the majority of the times, geotypical trees will be sufficient.

**Table 5-29: Component Selectors for GSFeature Dataset**

CS1	CS2	Component Name
001	001	Man-made point features
	002	Man-made point features class-level attributes
	016	Man-made point features extended-level attributes
	003	Man-made lineal features
	004	Man-made lineal features class-level attributes
	017	Man-made lineal features extended-level attributes
	005	Man-made areal features
	006	Man-made areal features class-level attributes
	018	Man-made areal features extended-level attributes
002	001	Natural point features
	002	Natural point features class-level attributes
	016	Natural point features extended-level attributes
	003	Natural lineal features
	004	Natural lineal features class-level attributes
	017	Natural lineal features extended-level attributes
	005	Natural areal features
	006	Natural areal features class-level attributes
	018	Natural areal features extended-level attributes
003	001	Trees point features
	002	Trees point features class-level attributes
	016	Trees point features extended-level attributes
004	001	Airport light point features
	002	Airport light point features class-level attributes
	016	Airport light point features extended-level attributes
	003	Airport lineal features
	004	Airport lineal features class-level attributes
	017	Airport lineal features extended-level attributes
	005	Airport areal features
	006	Airport areal features class-level attributes
	018	Airport areal features extended-level attributes
005	001	Environmental light point features
	002	Environmental light point features class-level attributes
	016	Environmental light point extended-level attributes

### 5.7.3.1 Default Read Value

Simulator client-devices should assume an empty tile when data is not available.

### 5.7.3.2 Default Write Value

The files associated with this dataset for area covered by tile at a given LOD need not be created if the source data is not available.



### 5.7.4 Tiled GTFeature Dataset

A GTFeature is a geotypical (point, lineal, or areal) feature whose optional modeled representation is also geotypical.

The GTFeature Dataset provides the position, size, orientation (points), shape (lineal and areals), type and attribution of point, lineal, and areal features. It is subdivided into the following components:

- CS1 = 001: Man-made features
- CS1 = 002: Trees features
- CS1 = 003: Moving model location features

Table 5-30: Component Selectors for GTFeature Dataset lists all of the components of the dataset. The allocation of CDB attributes to each of the components is prescribed by Table 5-27: Allocation of CDB Attributes to Vector Datasets.

The Moving model location features component is used to permanently position moving models (e.g., position a row of parked tanks or aircrafts on a runway). When referenced and positioned in this manner, these models cannot be moved and articulated during the simulation.

**Table 5-30: Component Selectors for GTFeature Dataset**

CS1	CS2	Component Name
001	001	Man-made point features
	002	Man-made point features class-level attributes
	016	Man-made point features extended-level attributes
	003	Man-made lineal features
	004	Man-made lineal features class-level attributes
	017	Man-made lineal features extended-level attributes
	005	Man-made areal features
	006	Man-made areal features class-level attributes
	018	Man-made areal features extended-level attributes
002	001	Tree point features
	002	Tree point features class-level attributes
	016	Tree point features extended-level attributes
	003	Tree lineal features
	004	Tree lineal features class-level attributes
	017	Tree lineal features extended-level attributes
	005	Tree areal features
	006	Tree areal features class-level attributes
	018	Tree areal features extended-level attributes
003	001	Moving Model location features
	002	Moving Model location features class-level attributes
	016	Moving Model location features extended-level attributes

#### 5.7.4.1 Default Read Value

Simulator client-devices should assume an empty tile when data is not available.

#### 5.7.4.2 Default Write Value

The files associated with this dataset for area covered by tile at a given LOD need not be created if the source data is not available.

### 5.7.5 Tiled GeoPolitical Feature Dataset

The GeoPolitical Feature dataset is used to provide information on the location, size and shape of abstract locations, lines and areas with respect to the surface of the earth. Generally speaking, the objects found in this dataset have no real-world physical representation (e.g., no physical characteristics such as mass) and correspond to abstract concepts (such as airspace, country boundary, danger zone).

The GeoPolitical Feature dataset is subdivided into the following components.

**Table 5-31: Component Selectors for GeoPolitical Feature Dataset**

CS1	CS2	Component Name
001	001	Boundary point features
	002	Boundary point features class-level attribute
	016	Boundary point features extended-level attribute
	003	Boundary lineal features
	004	Boundary lineal features class-level attribute
	017	Boundary lineal features extended-level attribute
	005	Boundary areal features
	006	Boundary areal features class-level attribute
	018	Boundary areal features extended-level attribute
002	001	Location point features
	002	Location point features class-level attribute
	016	Location point features extended-level attribute
	003	Location lineal features
	004	Location lineal features class-level attribute
	017	Location lineal features extended-level attribute
	005	Location areal features
	006	Location areal features class-level attribute
	018	Location areal features extended-level attribute
003	001	Constraint point features
	002	Constraint point features class-level attribute
	016	Constraint point features extended-level attribute
	003	Constraint lineal features
	004	Constraint lineal features class-level attribute
	017	Constraint lineal features extended-level attribute

CS1	CS2	Component Name
	005	Constraint areal features
	006	Constraint areal features class-level attribute
	018	Constraint areal features extended-level attribute

### 5.7.5.1 Boundary and Location Features

GeoPolitical Areal features are essentially used for closed surface related attributions whereas GeoPolitical Lineal features are used to model open areas as boundary delineations. When coarse specific locations are stored such as countries or cities, GeoPolitical Point features are used to locate the approximate geometric center of the related feature.

Some less-commonly used features could be 1) GeoPolitical Boundaries stored as Point features, or 2) GeoPolitical Locations stored as Lineals or Areal. The first case could be used to represent a simple boundary consisting of a single radius as given by the Point feature. The second case could be used to represent a city location with detailed Areal or Lineal vertices.

Figure 5-33: Example of International Boundaries, Figure 5-34: Example of City Locations, and Figure 5-35: Example of State Capital Locations and Time Zone Boundaries, all depict some sample cases where the GeoPolitical Feature dataset components can be used for.



**Figure 5-33: Example of International Boundaries**



Figure 5-34: Example of City Locations

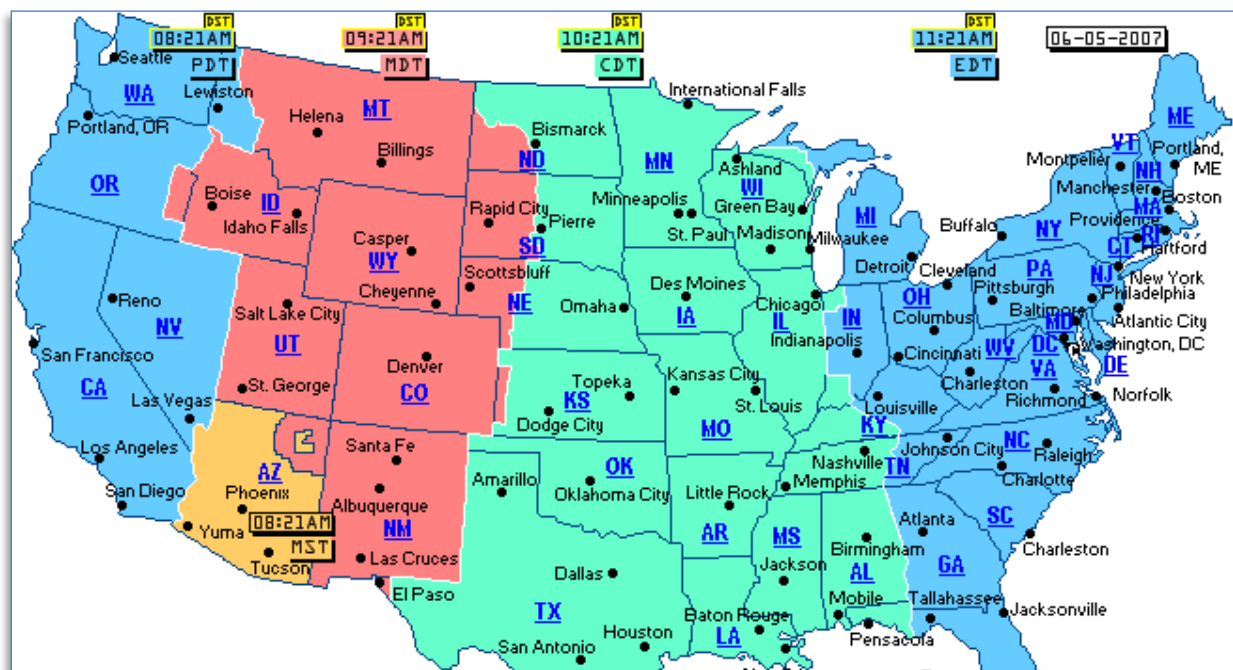


Figure 5-35: Example of State Capital Locations and Time Zone Boundaries

### 5.7.5.2 Elevation Constraint Features

There are many instances where modelers may wish to take advantage of the availability of position and altitude of cultural features in order to locally control the terrain elevation data at a point, along a specified contour line or within a given area.

This operation is usually performed off-line by the modeler and requires that the Elevation dataset be edited and re-generated offline.

In addition to this approach, the CDB Specification provides a Constraint Features component to the GeoPolitical Feature dataset, whose vector data is designed to instruct client-devices to *runtime-constrain* the Elevation dataset to a set of prescribed elevation values (thereby obviating the need to offline re-generate the Elevation dataset). At runtime, client-devices are required to apply the elevation constraints to the selected Terrain Elevation component of the Elevation dataset. The Constraint Features component provides modelers the ability to accurately control terrain elevation profiles even if the Elevation dataset consists in a uniform grid of modest resolution. Each of these features is associated with vertices which define elevation at the supplied geographic coordinates. This approach approximates Terrain Irregular Networks (TINs). The Constraint Features have the following Feature Attribute Codes (FACCs):

- ***Elevation Constraint Point (CA099-000):*** In the case of PointZ and MultiPointZ features, the points are used by the client-device to control the terrain elevation. The Feature's AHGT attribute must be set to TRUE. Shapefile features implemented as Point, PointM, MultiPoint, and MultiPointM are ignored.
- ***Elevation Constraint Line (CA099-001):*** In the case of PolyLineZ features, the lines are used by the client-device to control the terrain elevation. The Feature's AHGT attribute must be set to TRUE. Shapefile features implemented as PolyLine and PolyLineM are ignored.
- ***Elevation Constraint Area (CA099-002):*** In the case of PolygonZ and MultiPatch features, the areas are used by the client-device to control the terrain elevation. The Feature's AHGT attribute must be set to TRUE. Shapefile features implemented as Polygon and PolygonM are ignored.

The Data Dictionary of CDB Specification also makes provision for the representation of many hypsography features within the Geopolitical Dataset (e.g., contour lines, ridge lines, valley lines, spot elevation). By virtue of their semantics, these features have no associated modeled representation. The modeler can use these hypsography features to control the generation of the Terrain Elevation grid during the off-line CDB compilation process. This terrain constraining operation can be performed as the CDB is "assembled and compiled" by the SE tools. Note that runtime client-devices are not required to constrain the Terrain Elevation Dataset to hypsography features. When performed off-line, hypsography features must have AHGT set to True, thereby instructing the SE Tools to constrain the terrain elevation using the supplied (latitude, longitude, and elevation) coordinates. The Shapefile feature must be of type PointZ, MultiPointZ, PolyLineZ, PolygonZ, or MultiPatch.

While hypsography features can be used by the off-line tools to control the terrain skinning process, these features can be instead converted into Constraint Features, thereby deferring the terrain constraining process to runtime client-devices. This provides modelers the ability to accurately control terrain elevation profiles even if

the Elevation dataset consists in a uniform grid of modest resolution. In effect, the Constraint Features provides a storage-efficient means of capturing terrain contours without having to revert to high resolution terrain grids.

The application of constraint features to uniform and non-uniform gridded terrain elevation dataset is discussed in Appendix A.

#### 5.7.5.3 Default Read Value

Simulator client-devices should assume an empty tile when data is not available.

#### 5.7.5.4 Default Write Value

The files associated with this dataset for area covered by tile at a given LOD need not be created if the source data is not available.

### 5.7.6 Tiled RoadNetwork Dataset

The RoadNetwork Dataset is used to specify all of the road<sup>40</sup> networks. The points that make up the RoadNetwork lineals are primarily used to establish the road network topology. However, additional points can be used to more accurately define the actual path of the RoadNetwork lineals between each of the junction points of the network; alternately, points can be used to precisely specify the position of features along the RoadNetwork lineal. The altitude of each point is optional and specifies the ground level when present.

It is possible to associate an explicitly modeled representation (via the MODL and MODT attributes) to each RoadNetwork lineal. When provided, client-devices should instantiate the modeled representation for each point along the RoadNetwork lineal. Alternately, it is possible to specify a distinct modeled representation for each point along the RoadNetwork lineal feature by assigning a distinct RoadNetwork Figure Point to each point of the RoadNetwork lineal feature.

Table 5-32: Component Selectors for RoadNetwork Dataset lists all of the RoadNetwork Dataset components. The allocation of CDB attributes to each of the RoadNetwork dataset components is prescribed by Table 5-27: Allocation of CDB Attributes to Vector Datasets.

**Table 5-32: Component Selectors for RoadNetwork Dataset**

CS1	CS2	Component Name
001	011	Road/Airport Network Tile Connection 2D relationship
	015	Road/Airport Network Dataset Connection 2D relationship

<sup>40</sup> Within the context of the CDB, the term road encompasses a broad gamut of networks implemented as long, narrow stretch of smoothed or paved surfaces, made for traveling by foot, motor vehicle, carriage, etc., between two or more points. Examples of road networks are highways, interstates, roads, boulevards, streets, etc. It excludes railways.



CS1	CS2	Component Name
002	003	Road Network lineal features
	004	Road Network lineal features class-level attributes
	017	Road Network lineal features extended-level attributes
	007	Road Network lineal figure point features
	008	Road Network lineal figure point class-level attributes
	019	Road Network lineal figure point extended-level attributes
003	003	Airport Network lineal features
	004	Airport Network lineal features class-level attributes
	017	Airport Network lineal features extended-level attributes
	005	Airport Network areal features
	006	Airport Network areal features class-level attributes
	018	Airport Network areal features extended-level attributes

#### **5.7.6.1 Default Read Value**

Simulator client-devices should assume an empty tile when data is not available.

#### **5.7.6.2 Default Write Value**

The files associated with this dataset for area covered by tile at a given LOD need not be created if the source data is not available.

#### **5.7.7 Tiled RailRoadNetwork Dataset**

The RailRoadNetwork Dataset is used to specify all of the railroad<sup>41</sup> networks. The points that make up the RailRoadNetwork lineals are primarily used to establish the railroad network topology. However, additional points can be used to more accurately define the actual path of the RailRoadNetwork lineals between each of the junction points of the network; alternately, points can be used to precisely specify the position of features along the RailRoadNetwork lineal. The altitude of each point is optional and specifies the ground level when present.

It is possible to associate an explicitly modeled representation (via the MODL and MODT attributes) to each RailRoadNetwork lineal. When provided, client-devices should instance the modeled representation for each point along the RailRoadNetwork lineal. Alternately, it is possible to specify a distinct modeled representation for each point along the RailRoadNetwork lineal feature by assigning a distinct a RailRoadNetwork Figure Point to each point of the RailRoadNetwork lineal feature.

Table 5-33: Component Selectors for RailRoadNetwork Dataset lists RailRoadNetwork Dataset components. The allocation of CDB attributes to each of

---

<sup>41</sup> Within the context of the CDB, the term railroads encompasses a broad gamut of networks implemented as roads that are composed of parallel steel rails supported by ties and providing a track for locomotive-drawn trains or other wheeled vehicles.



the RailRoadNetwork dataset components is prescribed by Table 5-27: Allocation of CDB Attributes to Vector Datasets.

**Table 5-33: Component Selectors for RailRoadNetwork Dataset**

CS1	CS2	Component Name
001	011	RailRoad Network Tile Connection 2D relationship point
	015	RailRoad Network Dataset Connection 2D relationship point
002	003	RailRoad Network lineal features
	004	RailRoad Network lineal features class-level attribute
	017	RailRoad Network lineal features extended-level attribute
	007	RailRoad Network lineal figure point features
	008	RailRoad Network lineal figure point class-level attribute
	019	RailRoad Network lineal figure point extended-level attribute

#### 5.7.7.1 Default Read Value

Simulator client-devices should assume an empty tile when data is not available.

#### 5.7.7.2 Default Write Value

The files associated with this dataset for area covered by tile at a given LOD need not be created if the source data is not available.

### 5.7.8 Tiled PowerLineNetwork Dataset

The PowerLineNetwork Dataset is used to specify powerline<sup>42</sup> networks. The points that make up the PowerLineNetwork lineals are primarily used to establish the network topology. However, additional points can be used to more accurately define the actual path of the PowerLineNetwork lineals between each of the junction points of the network; alternately, points can be used to precisely specify the position of the poles, pylons, etc. of the PowerLineNetwork lineal. The altitude of each point is optional and specifies the ground level when present.

It is possible to associate an explicitly modeled representation (via the MODL and MODT attribute) to each PowerLineNetwork lineal. When provided, client-devices should instance the modeled representation for each point along the PowerLineNetwork lineal. Alternately, it is possible to specify a distinct modeled representation for each point along the PowerLineNetwork lineal feature by assigning a distinct a PowerLineNetwork Figure Point to each point of the PowerLineNetwork lineal feature. Allowed model representations are as per Section 5.7.1.4, Explicitly Modeled Representations. In that case of wired- or cabled-networks, the model must

<sup>42</sup> Within the context of the CDB, the term powerline encompasses a broad gamut of networks implemented through the use of wires, cables and/or pipes.



include the wire Attach Points<sup>43</sup>. Note also that client-devices are also required to automatically orient each instance of the modeled representation along the path of the PowerLineNetwork lineal. In the case where the entrance and exit angles are not identical, the orientation should be the average of both.

Table 5-34: Component Selectors for PowerLineNetwork Dataset, lists all of the PowerLineNetwork Dataset components. The allocation of CDB attributes to each of the PowerLineNetwork dataset components is prescribed by Table 5-27: Allocation of CDB Attributes to Vector Datasets.

**Table 5-34: Component Selectors for PowerLineNetwork Dataset**

CS1	CS2	Component Name
001	011	PowerLineNetwork tile connection 2D relationship point
	015	PowerLineNetwork dataset connection 2D relationship point
002	003	PowerLineNetwork lineal features
	004	PowerLineNetwork lineal features class-level attribute
	017	PowerLineNetwork lineal features extended-level attribute
	007	PowerLineNetwork lineal figure point features
	008	PowerLineNetwork lineal figure point class-level attribute
	019	PowerLineNetwork lineal figure point extended-level attribute

### 5.7.8.1 Default Read Value

Simulator client-devices should assume an empty tile when data is not available.

### 5.7.8.2 Default Write Value

The files associated with this dataset for area covered by tile at a given LOD need not be created if the source data is not available.

## 5.7.9 Tiled HydrographyNetwork Dataset

The HydrographyNetwork Dataset is used to specify hydrography<sup>44</sup> networks. The points that make up the HydrographyNetwork lineals are primarily used to establish the network topology. However, additional points can be used to more accurately define the actual path of the HydrographyNetwork lineals between each of the junction points of the network; alternately, points can be used to precisely specify the

<sup>43</sup> Note that wires are not explicitly modeled in the CDB. As a result, client-devices are required to automatically model them when rendering PowerLineNetwork lineal features. Such modeling can be achieved using a principle known as the catenary, which is the shape of a perfectly flexible chain suspended by its ends and which is characterized solely by the gravity action.

<sup>44</sup> Within the context of the CDB, the term hydrography encompasses a broad gamut of natural and man-made bodies of water such as oceans, lakes, rivers, canals, etc.

position of models of the HydrographyNetwork lineal. The altitude of each point is optional and specifies the ground level when present.

It is possible to associate an explicitly modeled representation (via the MODL and MODT attributes) to each HydrographyNetwork lineal. When provided, client-devices should instance the modeled representation for each point along the HydrographyNetwork lineal. Alternately, it is possible to specify a distinct modeled representation for each point along the HydrographyNetwork lineal feature by assigning a distinct a HydrographyNetwork Figure Point to each point of the HydrographyNetwork lineal feature. Allowed model representations are as per Section 5.3.1.4, Explicitly Modeled Representations.

Table 5-35: Component Selectors for HydrographyNetwork Dataset lists all of the components of the HydrographyNetwork dataset. The allocation of CDB attributes to each of the HydrographyNetwork dataset components is prescribed by Table 5-27: Allocation of CDB Attributes to Vector Datasets.

**Table 5-35: Component Selectors for HydrographyNetwork Dataset**

CS1	CS2	Component Name
001	011	HydrographyNetwork tile connection 2D relationship point
	015	HydrographyNetwork dataset connection 2D relationship point
002	003	HydrographyNetwork lineal features
	004	HydrographyNetwork lineal features class-level attribute
	017	HydrographyNetwork lineal features extended-level attribute
	005	HydrographyNetwork areal features
	006	HydrographyNetwork areal features class-level attribute
	018	HydrographyNetwork areal features extended-level attribute
	007	HydrographyNetwork lineal figure point features
	008	HydrographyNetwork lineal figure point class-level attribute
	019	HydrographyNetwork lineal figure point extended-level attribute
	009	HydrographyNetwork areal figure point features
	010	HydrographyNetwork areal figure point class-level attribute
	020	HydrographyNetwork areal figure point extended-level attribute

#### 5.7.9.1 Default Read Value

Simulator client-devices should assume an empty tile when data is not available.

#### 5.7.9.2 Default Write Value

The files associated with this dataset for area covered by tile at a given LOD need not be created if the source data is not available.



### 5.7.10 Tiled Vector Composite Material Table (VCMT)

The Vector Composite Material Table (VCMT) provides a list of the Composite Materials shared by all vector datasets. There is one VCMT for each CDB Geocell.

**Table 5-36: Vector Composite Material Table Component**

CS1	CS2	File Extension	Dataset Component Name	Dataset Component Description
Dataset 200, VectorMaterial				
001	001	*.xml	Vector Composite Material Table (VCMT)	The VCMT can be referenced by the CMIX attribute of the following datasets: <ul style="list-style-type: none"><li>• 100_GSFeature</li><li>• 101_GTFeature</li><li>• 102_GeoPolitical</li><li>• 201_RoadNetwork</li><li>• 202_RailRoadNetwork</li><li>• 203_PowerLineNetwork</li><li>• 204_HydrographyNetwork</li></ul>

#### 5.7.10.1 Data Type

The Vector Composite Material Table follows the XML syntax described in Section 2.5.2.2, Composite Material Tables (CMT).

#### 5.7.10.2 Default Read Value

The default values for the Vector Composite Material Table (Default\_Material\_Layer) can be found in \CDB\Metadata\Defaults.xml and can be provided to the client-devices on demand. If the default information cannot be found within the \CDB\Metadata\Defaults.xml file, the CDB Specification recommends defaulting to single substrate composite material whose base material is Default\_Base\_Material (BM\_LAND-MOOR).

#### 5.7.10.3 Default Write Value

The files associated with the Vector Composite Material Table for the area covered by a tile at a given LOD need not be created if the source data is not available. Tiles partially populated with data are not permitted.

## 5.8 Tiled Model Datasets

### 5.8.1 Tiled GSModel Datasets

Table 5-37 provides the component selectors for the GSModel datasets.

**Table 5-37: Component Selectors for GSModel Datasets**

CS1	CS2	File Extension	Component Name	Component Description
Dataset 300, GSModelGeometry Dataset 305, GSModelInteriorGeometry				
001	001..999	*.flt	Individual Geometry	One OpenFlight file containing the geometry of the shell or multiple files to represent the interior of a GSModel as described in Chapter 6.
001	001	*.zip	Geometry Archive	One archive regrouping individual model geometry OpenFlight files of a Tile-LOD.
Dataset 303, GSModelDescriptor Dataset 307, GSModelInteriorDescriptor				
001	001	.xml	Individual Descriptor	Provides the metadata associated with individual GSModels. See section 6.14, Metadata, for a description of the content.  <u>NOTE:</u> A model descriptor includes a Composite Material Table for the exclusive use by its corresponding model geometry datasets above.
001	001	.zip	Descriptor Archive	An archive of all model descriptors of a Tile-LOD.
Dataset 301, GSModelTexture Dataset 306, GSModelInteriorTexture				
-	-	*.rgb	Individual Texture	Individual base and subordinate textures applied on individual or tiled models of a Tile-LOD, see the complete list in section 5.3, CDB Model Textures.
001	001	*.zip	Texture Archive	An archive of all base and subordinate textures of a Tile-LOD.
Dataset 304, GSModelMaterial Dataset 308, GSModelInteriorMaterial				
001	001..255	*.tif	Composite Material Index	Each texel is an index into its corresponding GSModelCMT or GSModelInteriorCMT. Component selector 2 is the layer number.
002	001..254	*.tif	Composite Material Mixture	Each texel indicates the proportion (between 0.0 and 1.0) of the composite material found in the corresponding material layer. Component Selector 2 is the layer number. When the texels are of integral types, they are scaled to the range 0.0 to 1.0.
001	001	*.zip	Composite Material Archive	An archive of all layers of indices and mixtures of a Tile-LOD.
Dataset 309, GSModelCMT Dataset 311, GSModelInteriorCMT				
001	001	*.xml	Composite Material Table	Contains the definition of the composite materials referenced by the model material datasets above. Its format is as specified in section 2.5.2.2, Composite Material Tables (CMT)



CS1	CS2	File Extension	Component Name	Component Description
001	001	*.zip	CMT Archive	An archive of all CMTs of a Tile-LOD.

### 5.8.1.1 GSModel Archive Size Limit

The size of any GSModel archive is limited to 32 MB to permit reading, processing and writing the file at runtime.

### 5.8.2 Tiled T2DModel Datasets

Table 5-38 provides the component selectors for the T2DModel datasets.

**Table 5-38: Component Selectors for T2DModel Datasets**

CS1	CS2	File Extension	Component Name	Component Description
Dataset 310, T2DModelGeometry				
001	001	*.flt	Model Geometry	One OpenFlight file containing the geometry of the T2DModel of a Tile-LOD. The content of the file is explained in Chapter 6.
Dataset 312, T2DModelCMT				
001	001	*.xml	Composite Material Table	Contains the definition of the composite materials referenced by the model geometry dataset above. Its format is as specified in section 2.5.2.2, Composite Material Tables (CMT)

## Chapter 6

### 6 CDB OpenFlight Models

This portion of the CDB Specification defines a set of conventions to represent 2D and 3D models based on version 16.0 of the OpenFlight Scene Description Database Specification as annotated in Appendix C.

The conventions presented here address the needs of several types of simulation clients including OTW, FLIR, NVG, CGF, radar, and laser, acoustic, magnetic, visual and thermal sensors.

#### 6.1 OpenFlight File Header

The OpenFlight Header Record contains descriptive metadata about the manner vertices are encoded, and how these vertices are applied to an earth model and a projection type. On the other hand, the CDB Specification itself mandates a prescribed set of conventions in this regard; as a result, the following OpenFlight Header records must be set as follows:

- **Projection Type:**
  - Flat Earth, value 0: This is the type of projection used by most CDB Models, GTModel, GSModel, and MModel.
  - Geodetic, value 5: This is the type of projection used by T2DModels.
- **Earth Ellipsoid Model:**
  - WGS-84, value 0: This is the Earth model to use with T2DModels. The field is ignored with other CDB Models.

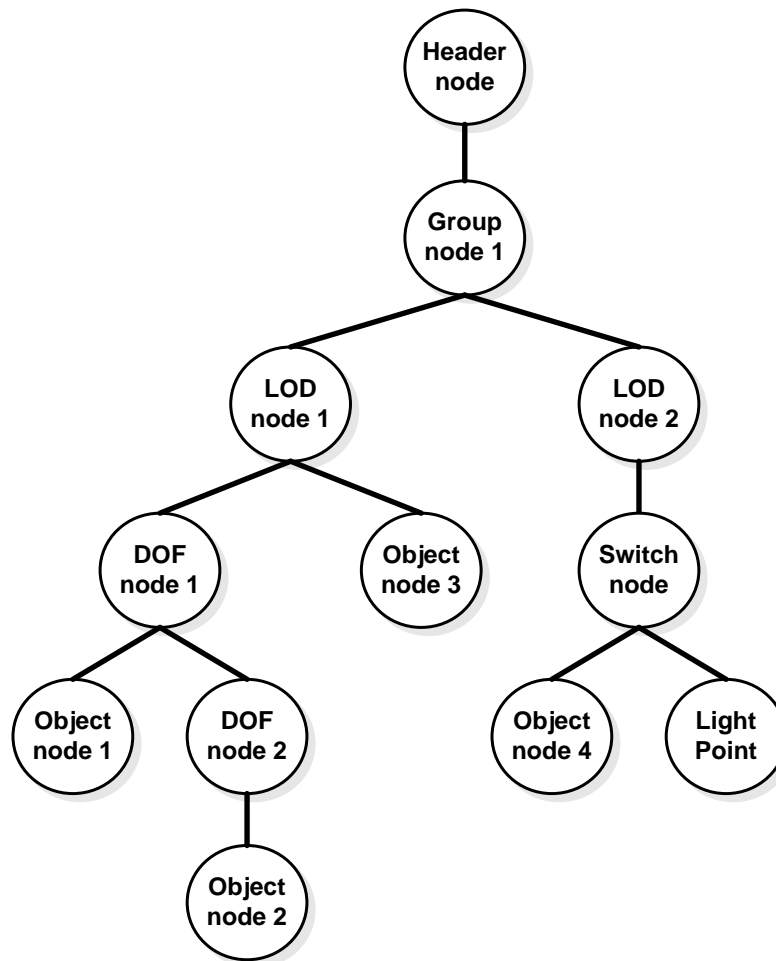
#### 6.2 OpenFlight Model Tree Structure

The internal structure of OpenFlight models is a tree structure that consists of nodes having child nodes as well as sibling nodes<sup>45</sup>. This type of tree structure is called a directed acyclic graph, or DAG. The general tree structure of a Model resembles that of Figure 6-1: General OpenFlight Tree Structure.

---

<sup>45</sup> In Appendix C, the section called *Database Hierarchy* explains in details how OpenFlight organizes its graph.





**Figure 6-1: General OpenFlight Tree Structure**

The CDB Specification uses Group Nodes to arrange Models in a hierarchical manner. This way of organizing models helps identify components of interest.

The CDB Specification refers to a number of OpenFlight nodes to store meaningful data for simulation client-devices. For a complete list of nodes supported by the CDB Specification, see Appendix C. The nodes listed here are the ones referred to by one or more CDB conventions.

An example of an ancillary record is the OpenFlight comment record. A comment record may appear once, anywhere after a node's primary record. The CDB Specification relies on comment records to extend the definition of OpenFlight nodes. Instead of using the Extension Record to create new primary and ancillary records, the CDB Specification uses comments to store the extra attributes required by the specialization of existing OpenFlight nodes<sup>46</sup>. Comment records were chosen over OpenFlight extensions in order to minimize any changes to the Creator tool or the

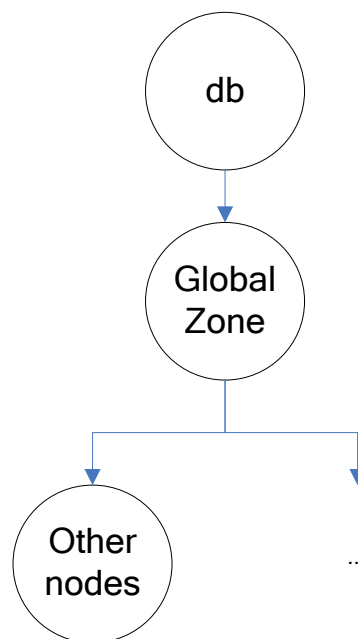
<sup>46</sup> CDB-compliant readers must ignore all OpenFlight extension records.

need to develop a plug-in to Creator. Using this approach, anybody can create CDB-compliant models using a plain version of Creator. Nevertheless, the development of Creator CDB plug-ins would improve modeler's efficiency. Such plug-ins could, for instance, offer a menu-based GUI to allow modelers to enter and edit CDB comments, while ensuring that syntax and conventions are fully adhered to.

The text contained in the comment record is formatted using the XML notation.

### 6.2.1 CDB Model Tree Structure

Based on Figure 6-1 above, the internal structure of CDB Models resemble this one:

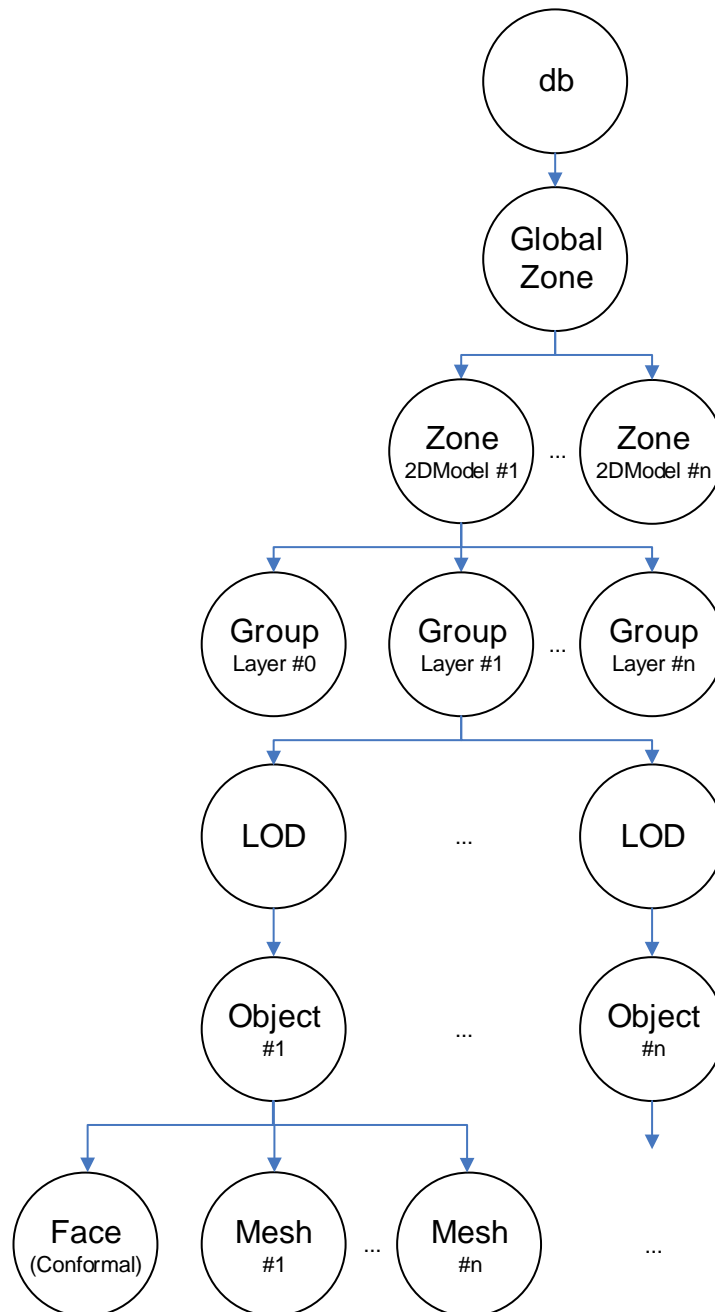


**Figure 6-2: Internal Structure of CDB Models**

All CDB Models have a global zone as their root node. This node identifies the model. Global zones are defined in section 6.5.2 below.

### 6.2.2 T2DModel Tree Structure

A T2DModel being a collection of 2DModels, each individual 2DModel occupies its own subtree of the graph. The general structure of a T2DModel is as follow:



**Figure 6-3: Internal Structure of T2DModels**

Each 2DModel is implemented as a Model Zone (section 6.5) with its own subgraph. 2DModels are disjoint and non-overlapping. A 2DModel is comprised of multiple layers; the layer number is expressed by the group's relative priority (section 6.3.4). Each layer has an optional LOD node followed by a fixed hierarchy of regular OpenFlight Object, Face, and Mesh nodes.

#### 6.2.2.1 Restrictions

T2DModels being an alternate representation of the terrain and its imagery and materials, a number of restrictions are necessary to ensure client devices can consume the dataset efficiently.

1. A 2DModel has at least two layers, layer 0 and layer 1.
2. Layer 0 is always empty because it represents the terrain on top of which subsequent layers are applied.
3. Each layer is composed of zero, one, or more OpenFlight Object nodes.
4. All Face and Mesh nodes share exactly the same set of graphic attributes (color, textures, material, and other flags). Stated differently, the Face and Mesh nodes provide the shape of the layer while the Object node controls its appearance.
5. Subfaces are not permitted because coplanar geometry is implemented through layers.

#### 6.2.2.2 Node Attributes

For T2DModels, node attributes (section 6.12) are permitted only at the zone levels; that is, at the global zone or at the individual 2DModel zones. Node attributes are not permitted at the Group, LOD, Object, Face, and Mesh node levels.

#### 6.2.3 The Use of Node Names

Although the CDB Specification defines naming conventions for objects stored in an OpenFlight file, the Specification does not constrain the OpenFlight node names themselves. Instead, the CDB Specification defines names that are assigned via XML tags stored in the comment record.

The following question arises, “Why not establish a set of CDB conventions around node names?” The answer lies primarily around constraints imposed by tools used to edit/create OpenFlight files. Tools such as Creator require unique node names throughout the OpenFlight file. The OpenFlight format Specification itself does not state that node names must be unique; however, a tool such as Creator prevents the modeler from entering the same node name twice.

To circumvent this limitation, the CDB Specification provides naming conventions through XML tags inserted in the comment record following a node’s primary record. This way of doing things leaves modelers with the needed freedom in naming nodes. The CDB Specification defines how to organize a model; all extra object attributes that do not fit in the current OpenFlight records are stored in the comment record, including object names.

Here is an example of XML tags stored in the comment record for a Group Node.

Table 6-1: Sample XML Tag Used in a Comment Record

```
<CDB:Zone  
  name = "zone name"  
>
```

All XML tags defined by the CDB Specification belong to a single XML namespace that is appropriately named CDB<sup>47</sup>.

#### 6.2.4 Model Master File

A Model Master file is an OpenFlight file that contains only external references to other OpenFlight files. The purpose of the master file is to ensure a Model is seen as a single “object” even though its constituents are stored in separate files. The use of a model master file provides a convenient means for modelers to reference all of the constituent files that make up the model. There is no other purpose associated to the master file.

As of version 3.1 of the Specification, the concept of a Model Master file is used in a single case, to regroup all representations (all LODs) of a geotypical model into a single OpenFlight file. However, the concept applies to all types of CDB Models. The concept can also be used to regroup a model’s shell with its interior. For this reason, expect new usage of the Model master file in future version of the Specification.

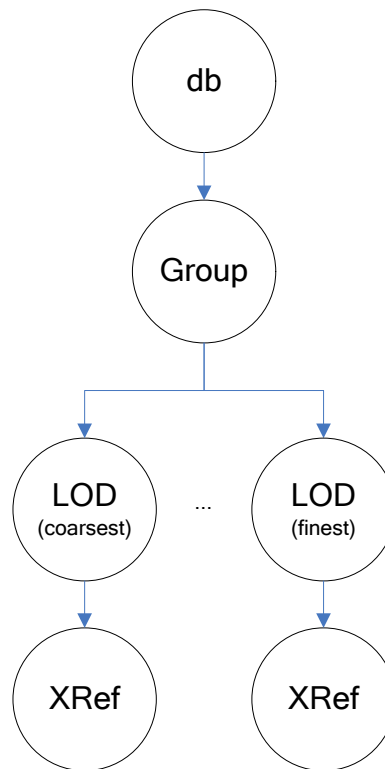
The master file is useful in two circumstances: when modelers create or edit Models, and when client devices want to discover at once all constituents of Models.

For modelers, it is useful (if not required) to edit a model using a single source file to present a coherent view of the model as a whole. For this reason, a master file with a set of LOD-XRef nodes is perfect to assemble and present a unified view of the model to edit.

Figure 6-4: Typical Structure of a Model Master File presents the general structure of a master file.

---

<sup>47</sup> The syntax to specify a XML namespace is `<ns:element>` where `ns` is the namespace and `element` is the XML element name (or simply tag).



**Figure 6-4: Typical Structure of a Model Master File**

The value found in the Significant Size field of the above LOD nodes matches the values found in Table 3-1: CDB LOD vs Model Resolution. The next section provides details on XRef nodes themselves.

## 6.2.5 Referencing Other OpenFlight Files

An OpenFlight external reference (XRef) node is used to refer to another OpenFlight file. The reference is made by specifying the filename and its path (absolute or relative). The CDB Specification requires that all references be made using a relative path. The XRef node (and its External Reference record) supports a number of options: Override flags, View-As-Bounding-Box flag, and Target Node Name. The CDB Specification supports none of these options.

Here are two cases to illustrate the use of XRef nodes.

### 6.2.5.1 Models Straddling Multiple Files

In the case of GTModels, GSModels, and MModels, the OpenFlight geometry can straddle multiple files. It could be seen in a moving model (e.g., a helicopter) whose pilot could be stored in a separate file. In that case, the file containing the pilot resides in the same directory as the file containing the helicopter itself. The helicopter would be stored in file 1:



```
\CDB\MModel\600_MModelGeometry\1_Platform\2_Air  
  \225_United_States\21_Utility_Helo\1_2_225_21_x_x_x\  
    D600_S001_T001_1_2_225_21_x_x_x.flt
```

The pilot would be stored in file 2:

```
\CDB\MModel\600_MModelGeometry\1_Platform\2_Air  
  \225_United_States\21_Utility_Helo\1_2_225_21_x_x_x\  
    D600_S001_T002_1_2_225_21_x_x_x.flt
```

The XRef node in file 1 would contain the following string:

```
.\D600_S001_T002_1_2_225_21_x_x_x.flt
```

where 1\_2\_225\_21\_x\_x\_x is the complete DIS code of the helicopter.

## 6.2.5.2 Models with Multiple Model-LODs

This is the case of the master file of a geotypical model. The master file (known as the GTModelGeometry Entry File) refers to all levels of details of the geometry files that reside in different sub-directories. Assuming a geotypical model representing a gothic church, the master file itself would reside in a directory such as this one:

```
\CDB\GTModel\500_GTModelGeometry\A_Culture\L_Misc_Feature  
  \015_Building\  
    D500_S001_T001_AL015_050_Church-Gothic.flt
```

The targets of the XRef nodes would all reside in directories such as these:

```
\CDB\GTModel\500_GTModelGeometry\A_Culture\L_Misc_Feature  
  \015_Building\Lnn\  
    D510_S001_T001_Lnn_AL015_050_Church-Gothic.flt
```

The resulting strings to use in the XRef nodes in the master file would resemble this:

```
.\Lnn\D510_S001_T001_Lnn_AL015_050_Church-Gothic.flt
```

where Lnn corresponds to the LOD the XRef file resides in.

## 6.3 Modeling Conventions

### 6.3.1 Model Coordinate Systems

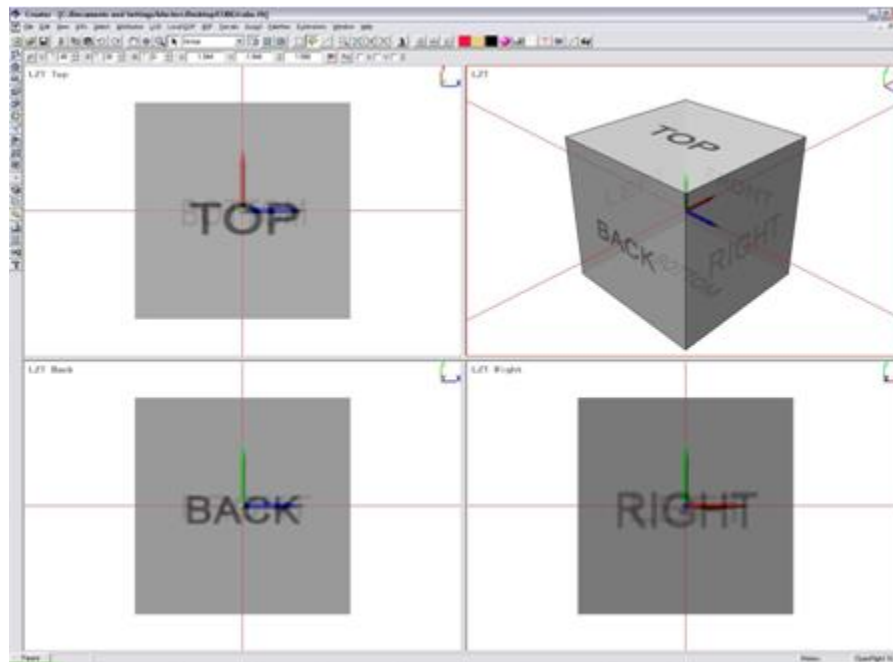
CDB Models use the same coordinate system convention as OpenFlight<sup>48</sup>. The X-axis traverses the model from left to right, the Y-axis goes from the back to the front and the Z-axis extends from the bottom to the top of the model. Figure 6-5: Model

---

<sup>48</sup> The DIS standard defines a different orientation for the axes of its coordinate system. DIS defines the X-axis as pointing to the front of the entity; the Y-axis pointing to its right and the Z-axis pointing down. Adoption of the DIS convention by the CDB Specification has been rejected due to the fact that the majority of models in existence have been created using tools such as Creator. The CDB Specification follows the same convention as the one used by these commercial tools. Note that if the CDB Specification were to follow the DIS convention, modelers would be required to create and edit their models upside down with respect to the reference plane provided by their tool.



Coordinate System is a screenshot<sup>49</sup> of Creator showing a CAD view of a semi-transparent cube.



**Figure 6-5: Model Coordinate System**

### 6.3.1.1 Origin

The location of the model origin is defined in the following manner:

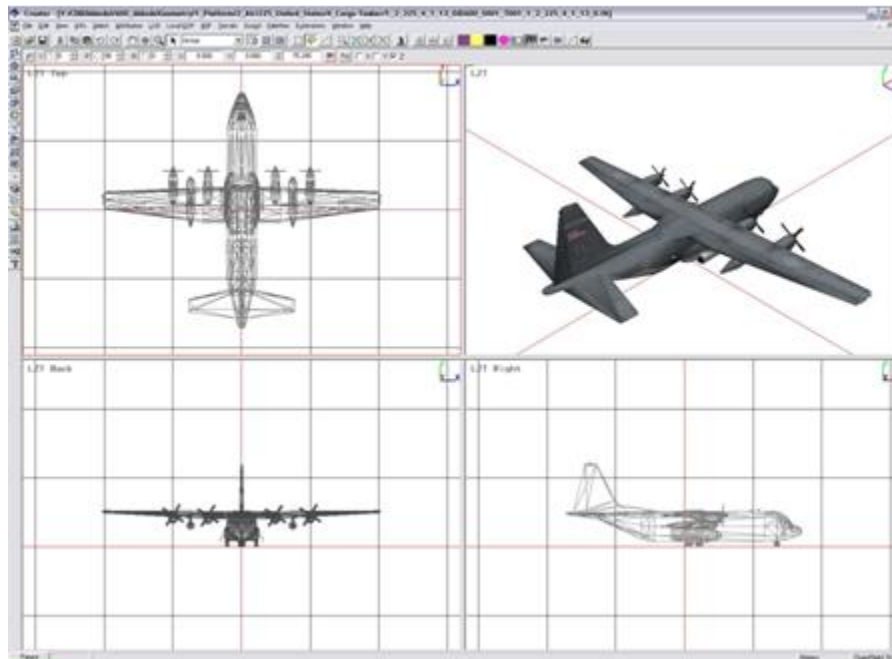
- In the XY plane, the origin must be located at the center of the bounding rectangle.
- Along the Z axis, the origin is selected to allow the model to be correctly positioned on ground for ground-related models, or on a water plan for surface and subsurface platforms.

The following examples will illustrate the above two rules.

1. Fixed wing aircraft – A KC-130 Hercules is illustrated below with its landing gears fully extended.

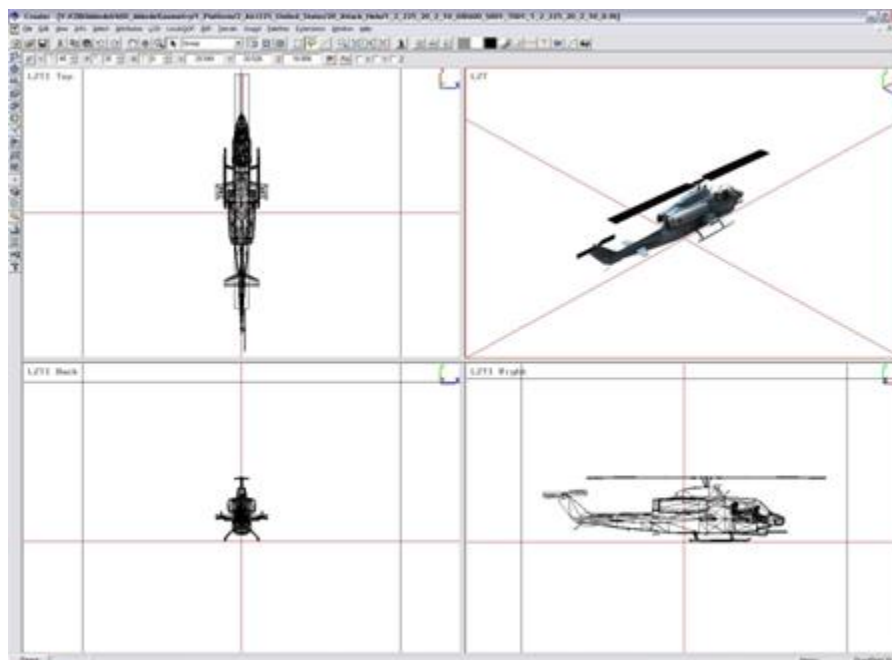
---

<sup>49</sup> Most figures in this chapter are actual screenshots from Creator.



**Figure 6-6: Coordinate System – Aircraft**

2. Helicopter – An AH-1W Super Cobra is shown below. Note that no equipment is mounted on its winglets.



**Figure 6-7: Coordinate System – Helicopter**

3. Surface ship – Shown below is the Arleigh Burke DDG-51 guided missile destroyer. Note how the XY plane defines the waterline.

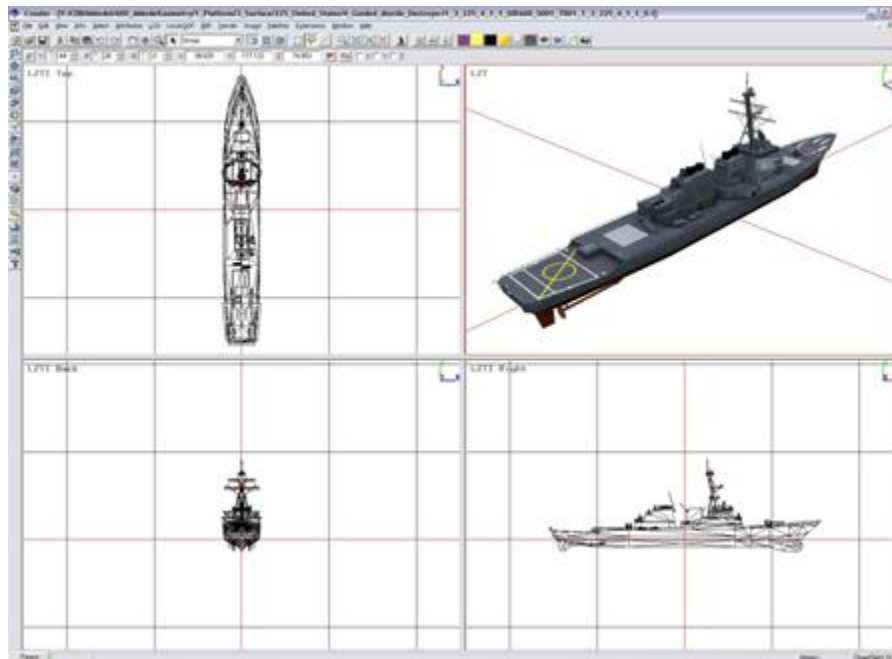


Figure 6-8: Coordinate System – Ship

4. Land Platform – The M1A2 Abrams is a main battle tank shown here with its desert skin.

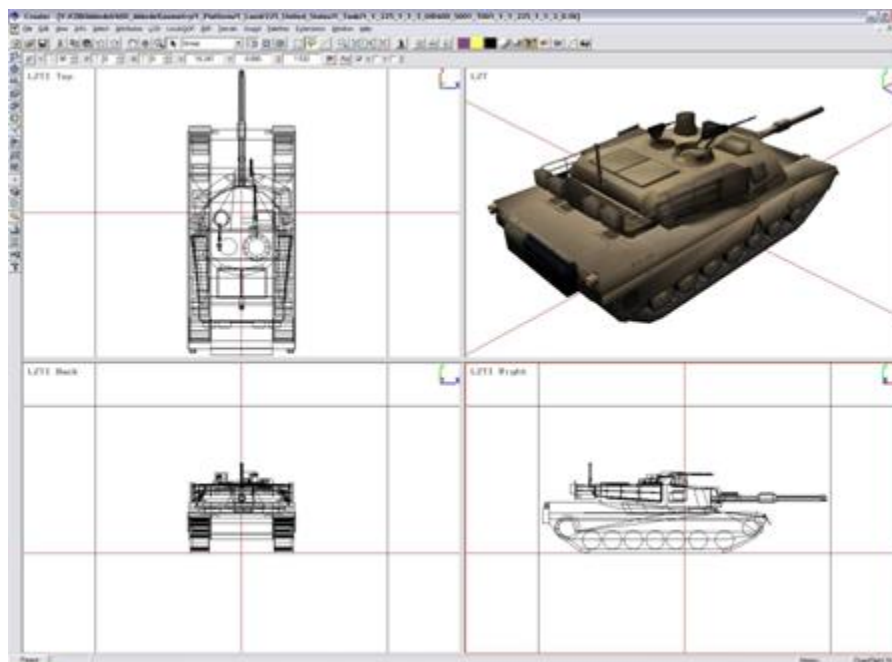
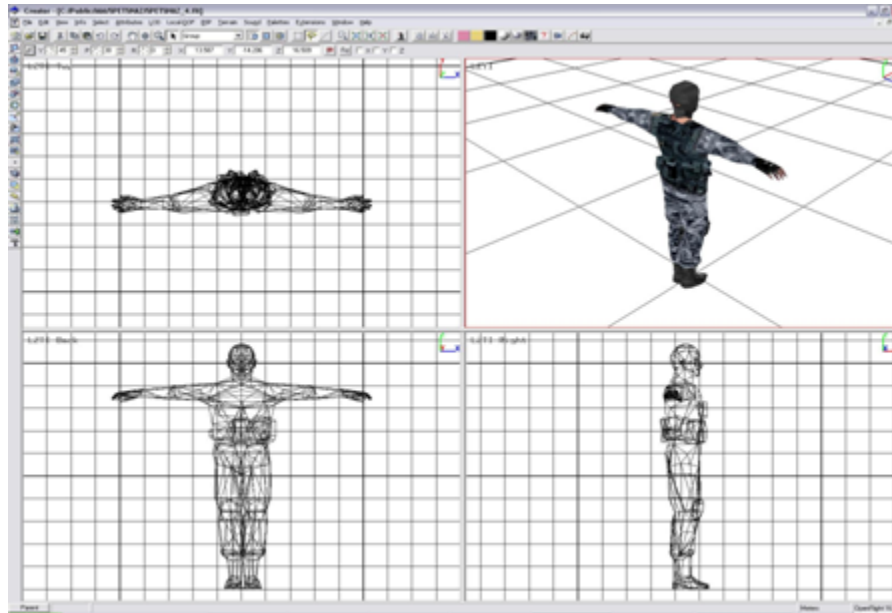


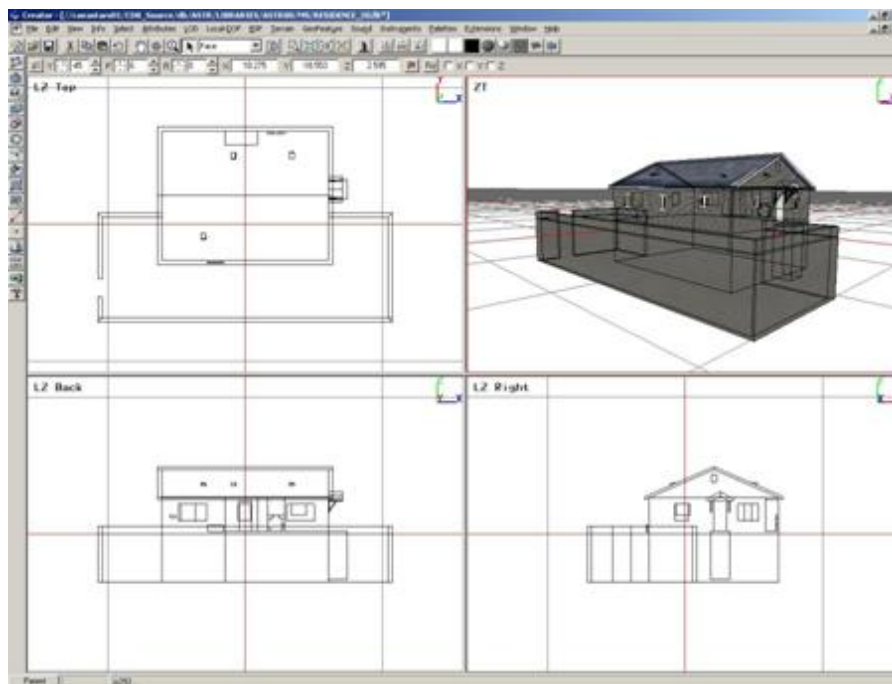
Figure 6-9: Coordinate System – Ground-based Model

5. Lifeform – A human lifeform (a soldier) is presented here.



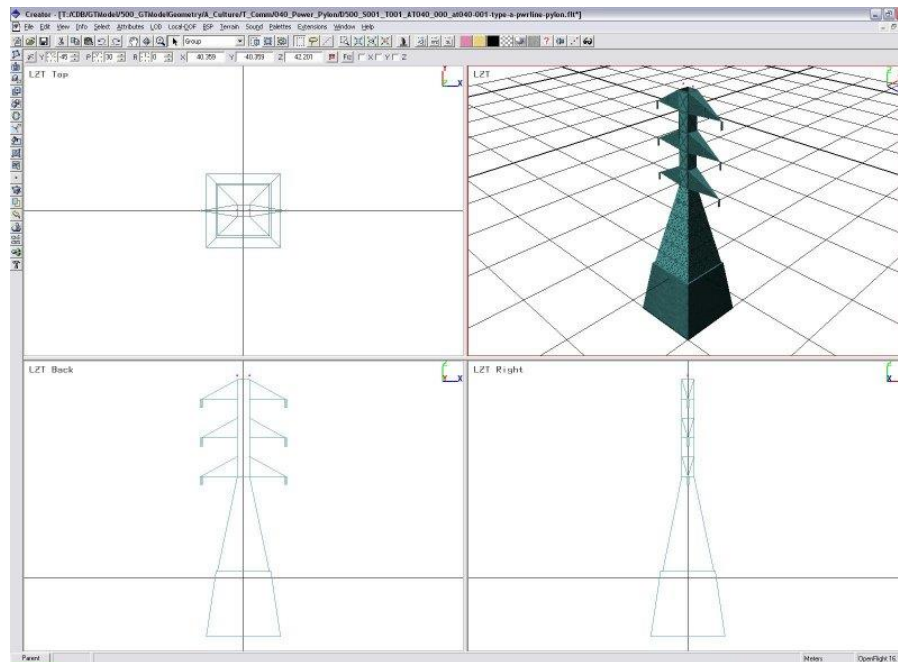
**Figure 6-10: Coordinate System – Lifeform**

6. Cultural Feature – When a Model represents a cultural feature, the origin is the point of insertion of the model into the ground. In general, the XY plane (at  $Z = 0$ ) delimits the basement from the first floor.



**Figure 6-11: Coordinate System – Cultural Feature**

7. Power Pylon – In the case of an electricity pylon, the front (and back) of the model is aligned with the general direction of the attached wires.



**Figure 6-12: Coordinate System – Power Pylon**

### 6.3.1.2 Local Coordinate Systems

Most OpenFlight nodes can have a local transformation used to create a local coordinate system. The origin and orientation of this coordinate system are expressed by a single transformation matrix.

When a transformation is specified for a node's primary record, only the matrix record (opcode 49) is considered by CDB client-devices. All other transformation records (opcode 76, 78, 79, 80, 81, 82 and 94) are discarded<sup>50</sup>.

### 6.3.1.3 Units

#### 6.3.1.3.1 GSModels and GTModels

The MODL attribute of the GSFeature and GTFeature datasets are used to reference GSModels and GTModels. In turn, the position of GSModels and GTModels are obtained from the coordinates of each point of the Shapefile; these coordinates are interpreted as the latitude (y), longitude (x), and elevation (z) coordinates that position the model within the CDB<sup>51</sup>.

<sup>50</sup> Typically, these transformations are used by modeling editing tools only.

<sup>51</sup> The local origin of the model is translated to the (lat, long) coordinates of the point feature. The elevation component is either obtained from the point feature (AHGT=True) or obtained from the elevation of the terrain at (lat, long) coordinates of

GSMODELS and GTMODELS are drawn to real-world dimensions using meters as their unit of measurement.

#### **6.3.1.3.2 MModels**

MModels are usually internally activated and positioned by the client-devices in response to a running simulation.

In the case of Moving Model location features, the MMDC attribute of the GTFeature dataset is used to reference a MModel. Otherwise, the MModel behaves exactly as a GTModel as described in section 6.3.1.3.1 above.

MModels are drawn to real-world dimensions using meters as their unit of measurement.

#### **6.3.1.3.3 T2DModels**

Vertex latitude (y) and longitude (x) coordinates are expressed in decimal degrees. The values are relative to the file's (implicit) origin which is the south-west corner of the tile. Note however, that the file's origin and size are implicitly defined by the tile position and the tile level-of-detail. The absolute position of each vertex is obtained by adding the vertex relative value to the tile origin.

T2DModels are used to model features that have no significant height with respect to the neighboring terrain; they are generally conformed to the terrain using the "Surface Conformal Mode" as explained in section 6.7, Model Conforming. Note that the vertices of T2DModels need not have Z-coordinate values that are always zero. For example, it is permissible to model a road lineal that is modestly elevated with respect to the neighboring terrain. Client-devices must be capable of handling T2DModels that are either perfectly surface-conformed to the terrain (all vertices have  $Z=0$ ) or modestly elevated (vertices with  $Z>0$ ) with respect to the terrain. Note that surface-conformed models with vertices with Z-coordinate values less than zero are, by definition, below the terrain.

#### **6.3.1.4 Roll, Pitch, Yaw**

Pitch, Roll and Yaw angles refer to rotations around the X, Y, and Z axes, respectively. Angles are measured in degrees. The Roll and Yaw angles vary from  $\pm 180$  degrees while the Pitch angle is limited to the range  $\pm 90$  degrees.

### **6.3.2 Geometry**

The CDB Specification assumes modelers adhere to the following set of constraints, rules and guidelines when creating the geometry of Models.

---

the point-feature (AHGT=False). The model's XY plane is rotated in accordance to the feature's AO1 attribute. The model's Z-axis is adjusted so that it is perpendicular to the WGS-84 earth model.



1. Create convex polygons. All lines joining any two points in the interior of the polygon also lie in the interior.
2. Create coplanar vertices. All of the vertices defining a polygon should lie in the same plane (required by virtually all rendering engines).
3. A polygon's front face is defined by a counter-clockwise ordering of its vertices.
4. Avoid T vertices. T-vertices occur when two or more adjacent polygons share an edge, and the polygons do not share a common vertex on that edge.
5. Avoid coplanar faces. A coplanar face is a polygon that lies directly on top of another polygon. Z-buffer fighting can occur when a Z-buffer system cannot resolve which polygon to display on top. The CDB Specification strongly recommends changing one of the coplanar faces to be a subface of the other in the hierarchy so that the client device such as an Image Generator can draw the faces in the correct order. An alternative is to use the Relative Priority field to implement layering (see section 6.3.4 below).
6. Favor mesh over individual polygons. The OpenFlight Mesh record allows the creation of triangle fans, triangle strips, quadrilateral strips, and indexed face sets. This construct is by far preferable to individual polygons built using the OpenFlight Face record.
7. Make use of instancing. Avoid repeating identical pieces of geometry. Create one object and repeat it by having multiple instances in different locations.

Users of the Presagis Creator modeling software should carefully read the document entitled *Creating Models for Simulation*, provided as part of the standard installation of the product. This document provides guidelines to build models geared toward real-time application.

### 6.3.3 Roof Tagging

OpenFlight has a provision for tagging polygons that are part of a roof. The Roofline flag can be found in both the Face and Mesh records. This flag is useful to apply a geospecific texture to the roof. When the flag is set, the client-device can discard the texture referenced by the polygon and use instead the geospecific texture that appears on the terrain.

More generally, the Roofline flag is used to tag any polygon whose texture can be replaced with a geospecific texture.

### 6.3.4 Relative Priority

The Relative Priority field appears in four OpenFlight primary records:

- Face Record
- Mesh Record
- Object Record
- Group Record





The field is required to implement layering, a method to handle coplanar geometry. By using the Relative Priority field, the modeler can construct more complex coplanar geometry than what is possible with subfaces.

Here is the definition of the field according to OpenFlight 16.0:

*Relative priority specifies a fixed ordering of the node relative to its sibling nodes. Ordering is from left (lesser values) to right (higher values). Nodes of equal priority may be arbitrarily ordered. All nodes have an implicit (default) relative priority value of zero.*

The CDB Specification further restricts the use of the Relative Priority field for complex coplanar geometry as follows. Using relative priorities, the modeler defines 'n' layers of coplanar geometry. Layers are numbered from 0 to 'n-1'. Layer 0, the base layer, must contain geometry that completely encompasses the geometry of subsequent layers. Other layers are processed in order, one after the other. A layer is made of one or more nodes; all nodes of a given layer have the same relative priority.

## 6.4 Model Identifiers

### 6.4.1 GSModel and GTModel Identifier

Although the name of the global zone of a GS or GT model is arbitrary, it is strongly suggested to use the value of their MODL attribute to name the global zone.

For instance, a GTModel stored in a file called

```
D500_S001_T001_AL015_004_Castle.flt
```

Would have its global zone identified like this:

```
<CDB:Zone  
  name="Castle"  
>
```

### 6.4.2 MModel Identifier

The MModel identifier is the common name of the moving model or the name of its part. The actual name is not covered by the Specification. An example of a MModel identifier is "M1A2" for the M1 Abrams tank whose DIS Entity Type is 1\_1\_225\_1\_1\_3\_0.

For instance, a moving model stored in a file called

```
D600_S001_T001_1_1_225_1_1_3_0.flt
```

Would have its global zone identified like this:

```
<CDB:Zone  
  name="M1A2"  
>
```

### 6.4.3 2DModel Identifier

A T2DModel itself does not need to be identified per se. However, it is required to identify individual 2DModels inside a T2DModel. This is done by assigning a unique zone name to each 2DModel. In addition, a 2DModel must have the same zone name across LODs and across tiles. Note that when a 2DModel is clipped by tile boundaries, each of the clipped model fragments will appear in distinct OpenFlight files of the T2DModel Dataset. When clipped, the 2DModel Identifier must appear once, in each of the T2DModel Tile-LODs. This is necessary to identify the parts of a 2DModel that straddle multiple Tile-LODs.

## 6.5 Model Zones

The concept of a model zone is of the utmost importance when creating models, particularly those used in military simulation applications.

A model zone represents a component<sup>52</sup> of interest on the Model. A model zone (as well as the component it represents) occupies a certain volume and is delimited by a bounding box. At least one simulator subsystem must be interested in a specific component to justify the creation of a corresponding zone. Examples of zones are a turret on a tank, or an engine on a platform, or an entrance door on a building, etc.

Since the model itself is of interest to the simulation, it will have at least one zone, the global zone. That will be the case for most Models used as cultural features; they will have a single zone. However, Models used as moving models will typically be subdivided into several zones.

To implement the concept of model zones, the CDB Specification uses the OpenFlight Group Node. Firstly, a Group Node can have child nodes to represent its own geometry as well as other zones. Secondly, a Group Node can have a bounding volume encompassing its child nodes and that can be used to represent the volume corresponding to the zone.

### 6.5.1 Definition

A Model Zone is an OpenFlight Group Node with a mandatory Bounding Box and the following XML tags in the comment field.

---

<sup>52</sup> A dictionary of CDB Component Names is provided in Appendix F.

**Table 6-2: XML Tags for Zones**

```
<CDB:Zone name="name" volume="closed|open">
... zone attributes
</CDB:Zone>
```

The zone name is mandatory. Remember that if the zone exists, it is because it has its importance for at least one client device. In general, all client devices interested in a zone will use its name to identify and control it.

The volume attribute is optional and specifies whether the zone represents a closed or open volume. By default, a zone represents a closed volume.

The following table lists the OpenFlight records required to represent a zone.

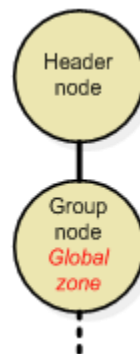
**Table 6-3: OpenFlight Records for a Zone**

```
GROUP
MATRIX (optional)
BOUNDING BOX (mandatory)
COMMENT (mandatory)
```

Note the use of the MATRIX record. It is necessary when the zone has a different position or orientation than its parent node. A zone can be thought of as a separate Model in itself. A zone has a natural orientation and its local coordinate system must indicate where their front, right, and back sides are. A zone is subject to the same convention as the model itself regarding the orientation of its coordinate system.

## 6.5.2 Global Zones

A CDB-compliant Model has at least one zone that encompasses the whole model and that is called the model global zone. The global zone is mandatory. Figure 6-13 illustrates the location of the global zone in the graph hierarchy.


**Figure 6-13: Model Global Zone**

The global zone contains the name of the Model contained in the OpenFlight file.

### 6.5.3 Zone Attributes

A Model Zone can have any number of attributes using the general mechanism described later in section 6.12, Model Attributes. However, two specific attributes are described here because of their particular relevance to the concept of Model Zone.

#### 6.5.3.1 Material

The Material attribute provides an indication of the principal material the zone is made of. Since the majority of man-made models are made of one principal material as well as several less important materials, it is strongly suggested to use the Material attribute in the model global zone to specify what that principal material is.

The Material attribute is an index into the Composite Material Table located within the Model Descriptor file described in section 6.14. The value of the Material attribute is a strictly positive integer. The syntax of the XML tag is:

```
<CDB:Zone>
  <Material> index </Material>
</CDB:Zone>
```

The Material attribute can also be assigned to OpenFlight Group and Object nodes. The syntax is the following:

```
<CDB:Group>
  <Material> index </Material>
</CDB:Group>

<CDB:Object>
  <Material> index </Material>
</CDB:Object>
```

For compatibility with version 3.1 and 3.0 of the Specification, a simplified (but deprecated) syntax is still supported for Object, Face, and Mesh nodes when the Material is the only attribute.

```
<CDB:Material>
  index
</CDB:Material>
```

#### 6.5.3.2 Temperature

When the zone has a heat source, such as an engine, it is known as a Hot Spot. The maximum temperature the zone can reach is specified using the Temperature attribute as shown here:

**Table 6-4: XML Tags for Hot Spots**

```
<CDB:Zone>
  <Temperature> maximum temperature </Temperature>
</CDB:Zone>
```

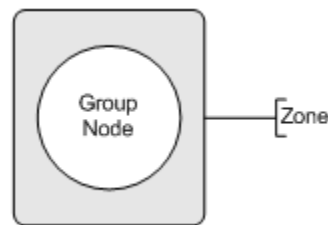
The temperature is expressed in Celsius degrees. Only integer values are permitted.

Appendix F supplies a comprehensive list of zones that are candidates for hot spots. Typical hot spot names are Engine and Chimney to name only these two. Other zones that are of interest for hot spots simulation are wings leading edge and other surfaces subjected to friction.

#### 6.5.4 Implementation Guidelines

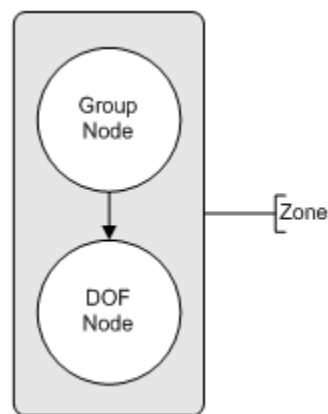
This section provides a set of guidelines to implement the concept of model zones. The guidelines provided here are also applicable to Model Points described in section 6.6.

A zone is made of at least one Group Node.



**Figure 6-14: Simple Zone**

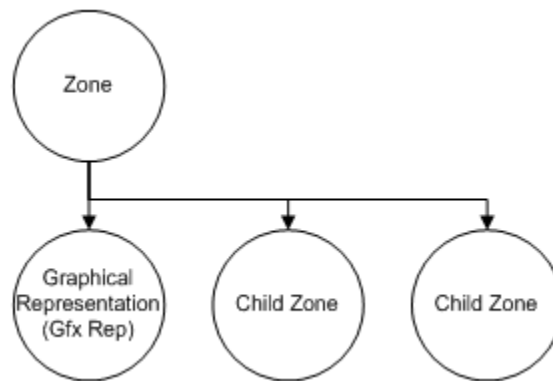
A zone may have an optional articulation by adding a DOF node.



**Figure 6-15: Articulated Zone**

To simplify the following diagrams, we will use a single circle to represent a zone, whether the zone is a single Group Node, or a pair of group and DOF nodes.

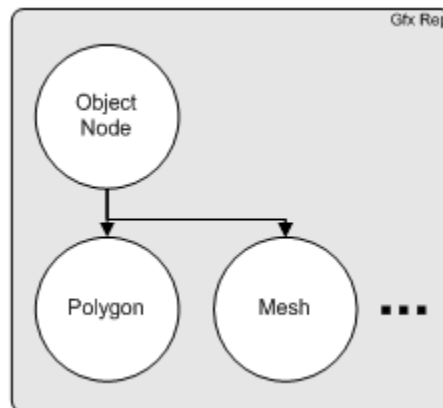
In general, a zone has a graphical representation as well as other child zones.



**Figure 6-16: Zone Hierarchy**

The graphical representation of a zone is itself subject to several possible implementations using various OpenFlight nodes.

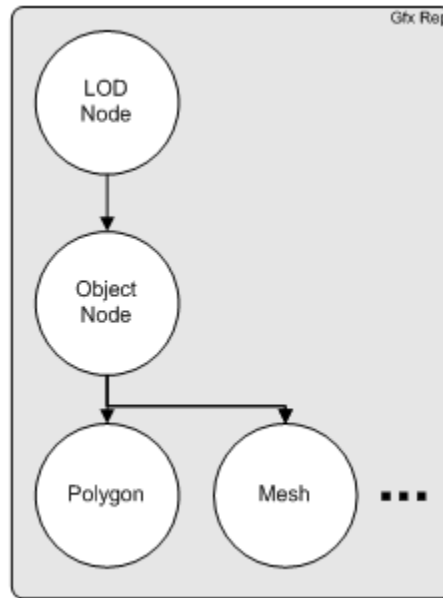
The simplest way to associate a graphical representation to a zone is to use an Object node with a combination of graphic primitives available in OpenFlight: polygons, triangles, quads, and meshes.



**Figure 6-17: Simple Zone Graphical Representation**

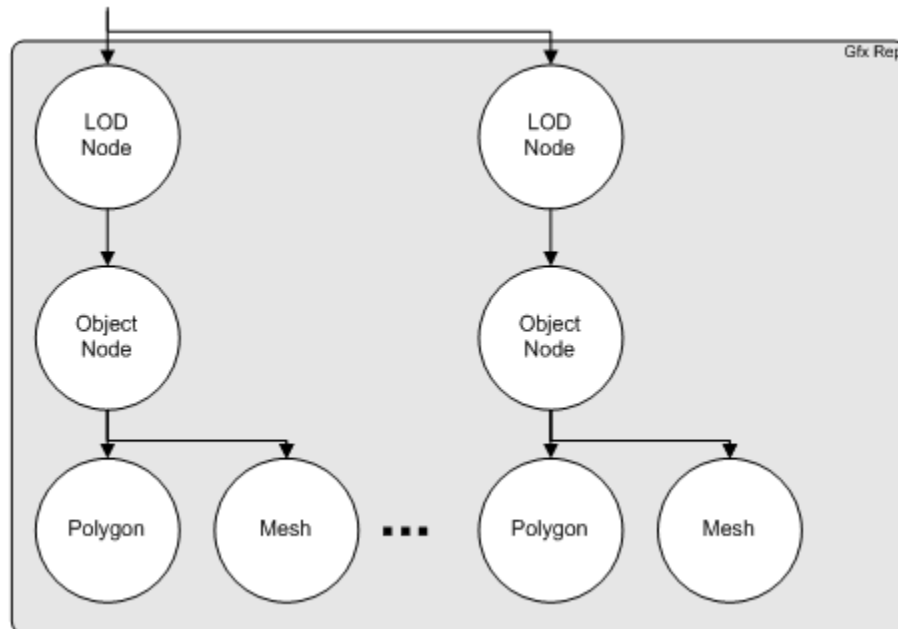
The modeler is also free to use a combination of LOD and Switch nodes to control the graphical representation of a zone.

For instance, an LOD node inserted before the object node is useful to inform the client device on how significant the graphical representation is.



**Figure 6-18: Additive LOD to Control the Graphical Representation**

If the modeler wants to provide two (or more) graphical representations for a zone, he should use two (or more) LOD nodes.

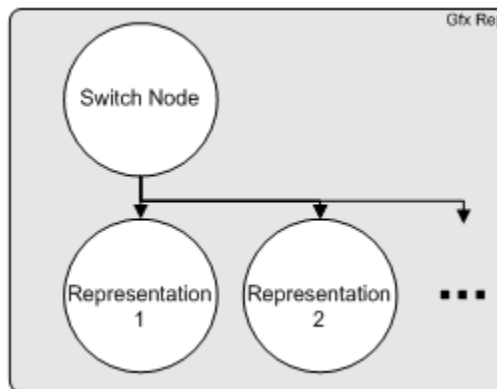


**Figure 6-19: Exchange LODs to Select the Graphical Representation**

Levels of details are discussed in length in Section 6.8, Model Levels-of-Detail.

If the modeler has several distinct graphical representations for the zone, he is also free to use a switch node to select between these representations.





**Figure 6-20: Switch Node to Select the Graphical Representation**

CDB Switches are discussed in depth in Section 6.9, Model Switch Nodes.

### 6.5.5 Model Zone Naming

A Model Zone Node is uniquely and unambiguously identified by concatenating with backslashes ('\') the names of all Model Zones traversed to reach it. When sibling CDB nodes have identical names, their name is appended with a sequence number in square brackets. Nodes are numbered starting with 1. Siblings are sorted in ascending order according to their X, Y, and Z coordinates. The leftmost sibling has the smallest XYZ coordinate while the rightmost sibling node has the largest XYZ coordinate. As a result, identical sibling CDB nodes are sorted from left to right (X-axis), then back to front (Y-axis), then bottom to top (Z-axis).

The following example provides a sample of Model Zone and Model Point names that would be used for a tactical fighter aircraft; the fighter has two pylons per wing, each pylon having 3 attach points. The resulting paths to each Model Zones and Model Points are as follow:

- \Fighter
- \Fighter\Wing[1]
- \Fighter\Wing[1]\Pylon[1]
- \Fighter\Wing[1]\Pylon[1]\Attach\_Point[1]
- \Fighter\Wing[1]\Pylon[1]\Attach\_Point[2]
- \Fighter\Wing[1]\Pylon[1]\Attach\_Point[3]
- \Fighter\Wing[1]\Pylon[2]
- \Fighter\Wing[1]\Pylon[2]\Attach\_Point[1]
- \Fighter\Wing[1]\Pylon[2]\Attach\_Point[2]
- \Fighter\Wing[1]\Pylon[2]\Attach\_Point[3]
- \Fighter\Fuselage
- \Fighter\Fuselage\Attach\_Point
- \Fighter\Wing[2]
- \Fighter\Wing[2]\Pylon[1]
- \Fighter\Wing[2]\Pylon[1]\Attach\_Point[1]

- \Fighter\Wing[2]\Pylon[1]\Attach\_Point[2]
- \Fighter\Wing[2]\Pylon[1]\Attach\_Point[3]
- \Fighter\Wing[2]\Pylon[2]
- \Fighter\Wing[2]\Pylon[2]\Attach\_Point[1]
- \Fighter\Wing[2]\Pylon[2]\Attach\_Point[2]
- \Fighter\Wing[2]\Pylon[2]\Attach\_Point[3]

Here is how to interpret some of these paths:

- The global zone is identified as \Fighter
- The left wing is \Fighter\Wing[1]
- The leftmost attach point is \Fighter\Wing[1]\Pylon[1]\Attach\_Point[1]
- The rightmost attach point is \Fighter\Wing[2]\Pylon[2]\Attach\_Point[3]
- There is a single attach point on the fuselage, \Fighter\Fuselage\Attach\_Point
- The inner pylon on the left wing is \Fighter\Wing[1]\Pylon[2]
- The inner pylon on the right wing is \Fighter\Wing[2]\Pylon[1]

## 6.5.6 Usages

### 6.5.6.1 Model Landing Zones

Landing zones are used primarily by the Computed Generated Forces (CGF) subsystem of the simulator. The landing zone information can be used during the set-up of mission scenarios since it provides CGF the location of known landing zones. Typically, landing zones are used to specify the location and dimension of helipads, aircraft carrier decks, etc. Appendix F lists several CDB Components that can act as landing zones.

Landing zones must have a bounding box that tightly fits the landing area. If required by the geometry of the landing zone, the modeler should create a local coordinate system that is axially oriented with the landing zone. Inserting a MATRIX record after the GROUP record does this.

The width and the length of the landing zone can be extracted by the client-device from the bounding box associated with the Group Node representing the zone. The landing zone geometry must be located under the Group Node to obtain meaningful dimensions.

### 6.5.6.2 Model Footprint Zones

A Model Footprint<sup>53</sup> conceptually represents the footprint (i.e., the terrain surface outline) of a model on the ground. The Model Footprint is modeled as a set of OpenFlight Face or Mesh records.

---

<sup>53</sup> The OpenFlight Face and Mesh records both have a flag called *Terrain Culture Cutout*. This flag is commonly designated as the *Cultural Footprint* flag within the simulation industry.

Client-devices are required to assume that the geometry that is associated with a Model Footprint Zone is hidden, regardless of the value of the OpenFlight *Hidden* flag of associated geometry. However, CDB content creation tools are nonetheless required to set the *Hidden* flag of the associated geometry<sup>54</sup>. The footprint geometry should be terrain conformed using the “Surface Conformal Mode” as explained in section 6.7, Model Conforming. This instructs client-devices to conform this footprint to the underlying terrain altimetry, regardless of its level-of-detail.

The Model Footprint is the set of polygons or meshes that result from the intersection of the model geometry with its XY plane<sup>55</sup>.

A polygon that is tagged as Model Footprint can be used by client-devices to identify the portion of the terrain that is covered by a model. The Cultural Footprint can be used by client-devices such as:

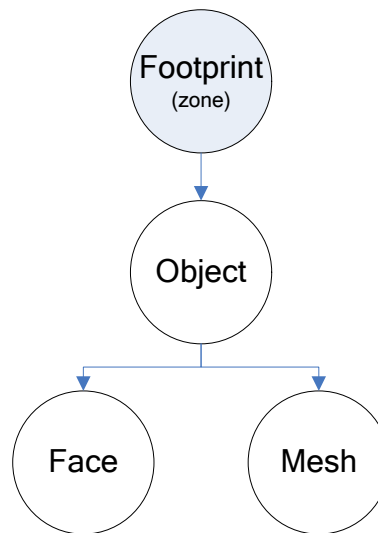
- Map generators that may not be interested in the full 3D geometry of a Model.
- Procedural SE generation software that may use model footprints to automatically extrude such footprints into 3D models.
- Simulation of ground-based SAF entities that would use model footprints to avoid collisions with features such as buildings and trees.

The CDB Specification requires that the Model Footprint be placed under a CDB Footprint Zone node. This CDB node facilitates the identification and discovering of footprints by client-devices. The subgraph representing the Footprint is presented here.

---

<sup>54</sup>This increases compatibility with OpenFlight readers that are not CDB-compliant.

<sup>55</sup> The Model Footprint polygon is not an absolute Z-positioned 3D polygon generated by the intersection of the model with the specific terrain it sits on - that would make the footprint of the model specific to that terrain. Furthermore, a different 3D polygon would be required for each possible terrain LOD.



**Figure 6-21: Footprint Zone Structure**

The Footprint zone is followed by an OpenFlight Object node with the necessary OpenFlight Face or Mesh nodes containing the footprint itself. All Face/Mesh nodes must have their *Terrain Culture Cutout* flag set. A Footprint zone is defined by the following XML tags.

**Table 6-5: Footprint Zone XML Tags**

<code>&lt;CDB:Zone name = "Footprint"/&gt;</code>
---

### 6.5.6.3 Model Cutout Zones

A Model Cutout Zone conceptually represents clipping geometry that is used to cut out the terrain from a 2DModel or a 3DModel. Cutouts are typically used in conjunction with model interiors and tunnels. The Model Cutout geometry defines a simple 3D convex volume (open or closed). A typical implementation of a Model Cutout Zone for a modeled building would consist of a simple cube. Similarly, the Model Cutout Zone for a tunnel entrance would consist of a simple cylinder or a partially-open cube (see Figure 5-9: Modeling of Wells, Overhanging Cliffs and Tunnels).

The Model Cutout is modeled as a set of OpenFlight Face or Mesh records. Client-devices are required to assume that the geometry that is associated with a Model Cut-Out Zone is hidden and cut-out. Client-devices should ignore the value of the OpenFlight *Hidden* and *Terrain Culture Cutout* flags of associated geometry. However, CDB content creation tools are required to set the *Hidden* and *Terrain Culture CutOut* flags of the associated geometry<sup>56</sup>.

<sup>56</sup>This increases compatibility with OpenFlight readers that are not CDB-compliant.

The Model Cutout geometry should be terrain conformed using the “Surface Conformal Mode” as explained in the 6.7, Model Conforming. This instructs the client-device to conform the Model Cutout to the underlying terrain altimetry, regardless of its level-of-detail.

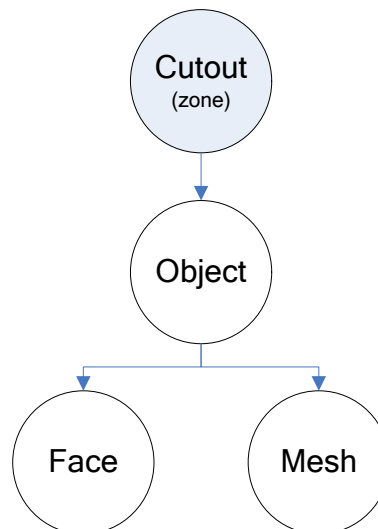
It is specified using the following XML syntax:

**Table 6-6: XML Tags for Landing Zones**

```
<CDB:Zone name="Cutout">
```

Polygons or meshes that are tagged as Model Cutout can be used by client-devices to identify the portion of the terrain that needs to be removed in order to reveal the interior of the model (say a building interior or a tunnel interior). The Model Cutout is necessary for models straddling the terrain surface and whose interior is modeled and viewed from within. The reason for this is that when the model is altitude-conformed onto the terrain, a hole must be cut into the terrain-LOD, so that the terrain itself does not visually interfere with the modeled building or tunnel interior.

The CDB Specification requires that the Cutout geometry be placed under a CDB Model Cutout Zone node. This CDB node facilitates the identification and discovery of model cutouts by client-devices. The subgraph representing the cutout is presented here.



**Figure 6-22: Cutout Zone Structure**

#### **6.5.6.4 Model Interior Zones**

This section focuses on how to represent the interior of Models for an intelligent use by client-devices.

A Model is composed of 2 parts: a shell, and an optional interior. The shell contains both the exterior and the pseudo-interior. Client-devices need only access the shell if

they do not need to penetrate and interact with the interior of the models; otherwise, they require both the shell and the interior. The shell of a Model is stored in five (5) datasets:

- ModelGeometry
- ModelTexture
- ModelSignature
- ModelDescriptor
- ModelMaterial

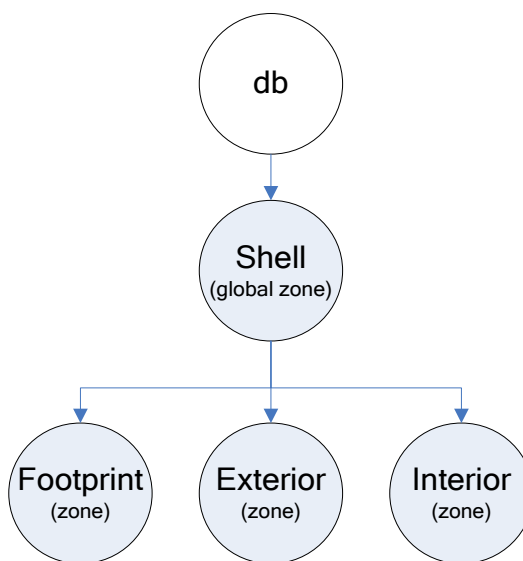
The optional model interior is stored in four (4) datasets:

- ModelInteriorGeometry
- ModelInteriorTexture
- ModelInteriorDescriptor
- ModelInteriorMaterial

Refer to appendix A.5 for guidelines on Handling of Model Interiors.

#### 6.5.6.4.1 Model Pseudo-Interior Zone

The pseudo-interior is the portion of the shell that contains geometry also represented in the interior dataset. This geometry represents what is visible from the outside and is necessary to ensure the integrity and completeness of the shell. Since the pseudo-interior is a placeholder for the real interior, it must be placed under its own subgraph and identified by a CDB zone whose name is “Interior”.



**Figure 6-23: Model Shell Structure**

The name “Interior” is a reserved component name allowing a client-device to identify the node that is to be replaced by an entire dataset, namely the

ModelInteriorGeometry dataset. The pseudo interior is mutually exclusive with the real interior defined in section 6.5.6.4.2, Model Interior Zone, below.

Figure 6-23 also illustrates how to structure the shell of a Model that has a real interior. The model is divided in three components: the footprint of its exterior, the geometry of its exterior, and the geometry of its pseudo-interior. Therefore the names of these three components are “Footprint”, “Exterior”, and “Interior” as illustrated by the following XML tags.

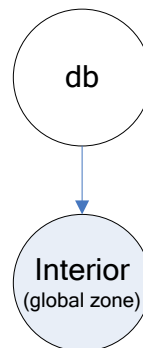
**Table 6-7: Shell Zones XML Tags**

```
<CDB:Zone name = "Footprint"/>
<CDB:Zone name = "Exterior"/>
<CDB:Zone name = "Interior"/>
```

Footprints were discussed earlier in section 6.5.6.2, Model Footprint Zones.

#### 6.5.6.4.2 Model Interior Zone

The Model interior itself must have a global zone whose name is “Interior”. Accordingly, the graph of the interior of the model will present the following structure. Note that real interior must not include the modeled representation of the shell.



**Figure 6-24: Model Interior Structure**

The Interior zone contains one or more floors as well as the partitions separating these floors. An Interior zone is defined by the following XML tags.

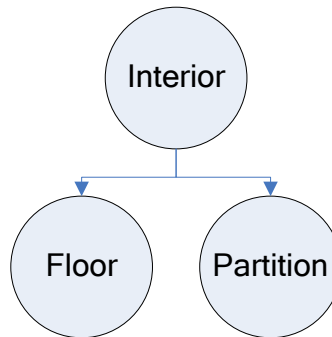
**Table 6-8: Interior Zone XML Tags**

```
<CDB:Zone name = "Interior">
  <Ground_Floor> index </Ground_Floor>
</CDB:Zone>
```



The <Ground\_Floor> is optional. It contains the index of the Floor that represents the ground floor of the model interior. By default, the ground floor is floor number 1.

The subgraph representing the Interior zone has the following structure.



**Figure 6-25: Interior Zone Structure**

The Interior zone has two (2) kinds of child nodes: Floor and Partition. The Interior has at least one Floor. When the Interior has several Floors, the separating Partitions appear as siblings of the Floor nodes. These Partitions contain external Apertures that connect two Rooms on different Floors. These external Apertures are later referenced by Rooms.

#### **6.5.6.4.2.1 Model Interior Topology**

To navigate through the interior of Models, simulator client-devices need to know the connections between the elements composing the interior, such as floors, rooms, doors, or fixtures. In addition, these elements must be identified and attributed for use by computer generated forces (CGF) client-devices. For this reason, the CDB Specification has opted for reuse and adoption of version 2 of the UHRB specification<sup>57</sup>.

The CDB Specification maps the UHRB Object Model to the OpenFlight Scene Graph using the concept of CDB nodes.

The UHRB object model proposes twelve (12) classes. Of these, four (4) are abstract base classes and one (1) is a provision for future expansion of the UHRB specification. The remaining seven (7) concrete classes are mapped to CDB Zone nodes. The UHRB Class Names and their corresponding CDB Zone Names are:

<sup>57</sup> See references [35] and [36] in section 10, Reference Documents.

**Table 6-9: UHRB Class Names and corresponding CDB Zone Names**

UHRB Class Name	CDB Zone Name
UHRB_TEMPLATE	Interior
UHRB_FLOOR_LEVEL_COMPONENT	Floor
UHRB_SURFACE_COMPONENT	Surface
UHRB_ROOM_COMPONENT	Room
UHRB_FIXTURE_COMPONENT	Fixture
UHRB_APERTURE_COMPONENT	Aperture
UHRB_FIXED_PARTITION_COMPONENT	Partition

The above CDB nodes are treated the same way as any other CDB nodes. In particular, Floor, Room, Partition, Aperture, Fixture, and Surface nodes are numbered following the conventions established in section 6.5.5, Model Zone Naming; they also have zone attributes such as the Material Index.

#### 6.5.6.4.2.2 Model Interior Topology Attributes

This section describes the CDB mechanism that expresses the possible connections between compartments and apertures. The definition of a CDB Zone is extended with the addition of one XML tag indicating which other components are connected to this one.

The following table presents the revised syntax of the XML tags defining a CDB Zone. The addition is highlighted in yellow.

**Table 6-10: XML Tags for Zone Connections**

```
<CDB:Zone name="name" volume="closed|open">
  <Material> index </Material>
  <Temperature> value </Temperature>
  <ConnectTo> path </ConnectTo>
  ...
</CDB:Zone>
```

The <ConnectTo> tag may appear zero or more times, allowing for the definition of any number of connections to other components. The other tags (Material and Temperature) retain their current definition. In particular, the use of the <Material> tag is encouraged to define the material the components are made of.

The presence of the <ConnectTo> tag is restricted to a set of three (3) components: global zone, compartments and apertures. A connection is unidirectional; it goes from the zone that contains the <ConnectTo> tag to the zone referenced by the

path. The path is either relative or absolute. When a relative path is used, it identifies a sibling of the current zone. Here are some path examples.

**Table 6-11: Examples of Absolute and Relative Paths**

```

Example 1:
<CDB:Zone name="Interior">
  <ConnectTo> \Interior\Section[1]\Level[1]\Aperture[5] </ConnectTo>
</CDB:Zone>

Example 2:
<CDB:Zone name="Aperture[5]">
  <ConnectTo> \Interior\Section[1]\Level[2]\Compartment[3] </ConnectTo>
</CDB:Zone>

Example 3:
<CDB:Zone name="Compartment[3]">
  <ConnectTo> Aperture[1] </ConnectTo>
  <ConnectTo> \Interior\Section[1]\Level[1]\Aperture[5] </ConnectTo>
</CDB:Zone>

```

Example 1 is an absolute path, expressed as a directory name, starting with the topmost zone, the global zone. It tells us that there is one entrance into the model interior through the fifth aperture (Aperture[5]) on the first level (Level[1]) of the first section (Section[1]) of the model interior (\Interior).

Example 2 is also an absolute path. It tells us that the fifth aperture (Aperture[5]) has a single connection to the third compartment (Compartment[3]) of the second level (Level[2]) of the first section (Section[1]) of the model interior (\Interior).

Example 3 illustrates how to use a relative path. It tells us that the third compartment (Compartment[3]) has two exits. The first exit is through the first aperture (Aperture[1]) of the current level. The second exit is through the fifth aperture (Aperture[5]) on the first level (Level[1]) of the first section (Section[1]) of the model interior (\Interior).

Example 1 tells us to use Aperture 5 to enter into the model interior. Example 2 further informs us that Aperture 5 brings us into Compartment 3. Example 3 says that we can exit Compartment 3 through either Aperture 1 or 5.

The global zone (the top level node) node provides the list of apertures representing entrances into the model. If the global zone does not provide at least one aperture to enter the model, then the model interior is unreachable. To exit a compartment, it must connect to at least one aperture; if not, you may be able to enter the compartment, but you will not be able to exit. Finally, an aperture allows entrance into compartments. An aperture without connection is an exit point; in that case, a compartment must connect to the aperture.

Appendix J presents the XML schema governing the construction of a valid CDB Zone. The schema includes provision for the <ConnectTo> tag.

#### 6.5.6.4.3 Floor Zone

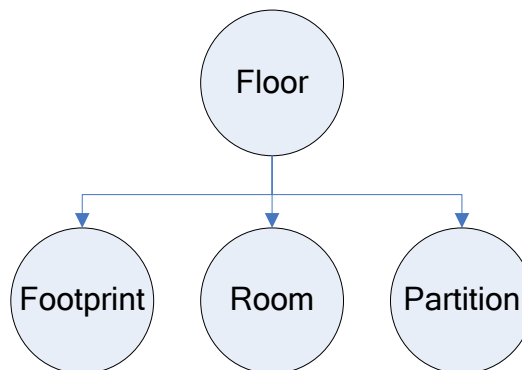
A Floor zone contains one or more Rooms, and all Partitions shared by these Rooms. A Floor is defined by the following XML tags.

**Table 6-12: Floor Zone XML Tags**

```
<CDB:Zone name = "Floor">  
  <Label> floor name </Label>  
</CDB:Zone>
```

The <Label> is optional. It can be used to give the Floor a meaningful name such as “Ground Floor”, “Basement”, “Mezzanine”, or “Penthouse”.

The subgraph representing a Floor has the following structure.



**Figure 6-26: Floor Zone Structure**

The Footprint of a Floor is the minimum enclosing polygon containing all of the footprints of all of the Rooms on the Floor as well as the footprints of all of the Partitions associated with those Rooms. The Footprint is defined as per section 6.5.6.2, Model Footprint Zones. The Partitions contain the Apertures that connect two Rooms together. These Apertures are later referenced by Rooms.

#### 6.5.6.4.4 Room Zone

A Room zone owns all its Surfaces and may contain Fixtures. A Room is defined by the following XML tags.

**Table 6-13: Room Zone XML Tags**

```

<CDB:Zone name = "Room">
  <Label> room name </Label>
  <Aperture> path to aperture 1 </Aperture>
  ... other apertures as needed
  <Partition> path to partition 1 </Partition>
  ... other partitions as needed
</CDB:Zone>

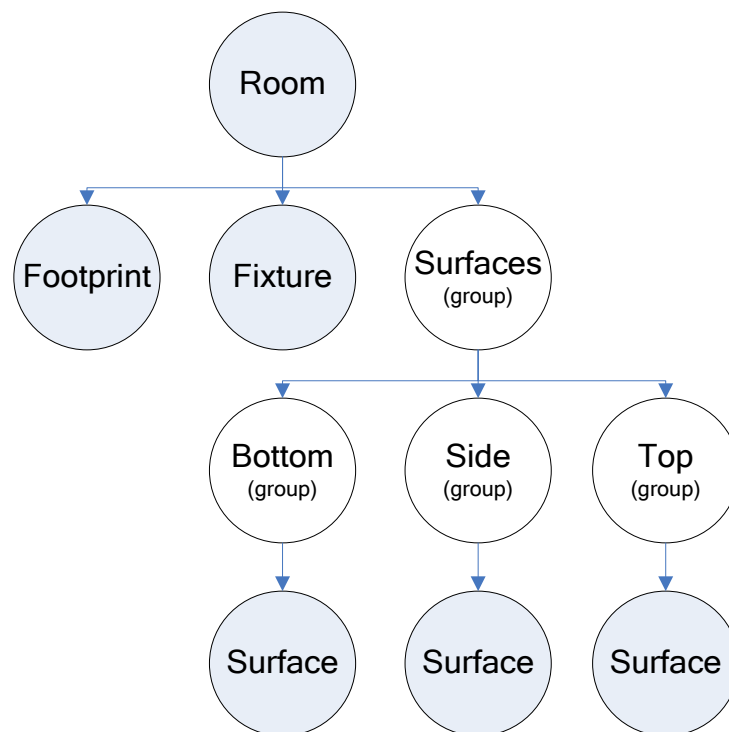
```

The <Label> is optional. It can be used to better identify the Room by its usual name. Examples are cubicle, toilet, conference room, atrium, office, electrical room, janitor room, etc.

The <Aperture> is optional but is likely to appear at least once, unless the Room is permanently closed and cannot be accessed. It points to one Aperture that connects this Room with another Room on this Floor or on another Floor. Two Rooms on the same Floor are connected through an Aperture in a Partition on the current Floor. Two Rooms on two different Floors are connected through an external Aperture in a Partition from the Interior zone. The path to an Aperture is built as specified in section 6.5.5, Model Zone Naming.

The <Partition> is also optional, but again, is likely to appear several times since a typical Room has a floor, a set of walls, and a ceiling.

The subgraph representing a Room has the following structure.


**Figure 6-27: Room Zone Structure**

The footprint is the smallest polygon containing all of the bottom surfaces when projected onto the XY plane<sup>58</sup>. There can be zero or more Fixtures in a Room. The Surfaces making up the volume of the Room are separated in three (3) groups (Bottom, Side, and Top) as defined by the UHRB specification.

#### 6.5.6.4.5 Fixture Zone

A Fixture zone is defined in the same manner as a Room; it is made of a number of Surfaces defining a closed volume. The Fixture is defined by the following XML tags.

Table 6-14: Fixture Zone XML Tags

```
<CDB:Zone name = "Fixture">  
  <Label> fixture name </Label>  
  <Moveable> true/false </Moveable>  
</CDB:Zone>
```

The <Label> is optional. It allows the modeler to describe what this fixture represents.

The <Moveable> flag is optional. It indicates whether or not the Fixture can move or if it is fixed. By default, the fixture does not move; if it does, the flag is set to true. A piece of furnitures is an example of moveable fixture while a column is an example of a fixed one.

The subgraph representing a Fixture is similar to that of a Room, except for the need to differentiate between the kinds of Surfaces. Its structure is presented here.

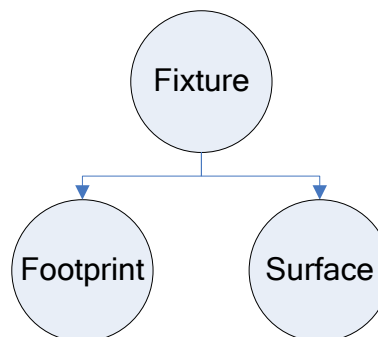
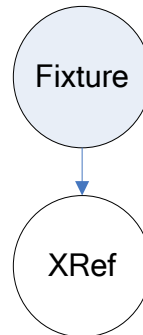


Figure 6-28: Fixture Zone Structure

The Footprint is the smallest polygon containing all of the Surfaces when projected onto the XY plane. The Surfaces form a closed volume, meaning there is no hole in the Fixture.

<sup>58</sup> This definition of a room footprint comes from the UHRB Specification.

Alternately, to permit reuse of common fixtures stored in the GTModel Library, the Fixture may reference an existing model through the use of an XRef node. In that case, the following subgraph is to be used.



**Figure 6-29: Fixture Zone Structure**

#### 6.5.6.4.6 Partition Zone

A Partition zone has Apertures, makes reference to all Surfaces composing it, and refers to its adjacent Rooms. The Partition is defined by the following XML tags.

**Table 6-15: Partition Zone XML Tags**

```

<CDB:Zone name = "Partition">
  <Label> partition name </Label>
  <Room> path to adjacent room 1 </Room>
  <Room> path to adjacent room 2 </Room>
  <Surface> path to surface 1 </Surface>
  ... other surfaces as needed
</CDB:Zone>
  
```

The <Label> is optional. It allows the modeler to better identify the type of Partition: wall, floor, ceiling, etc.

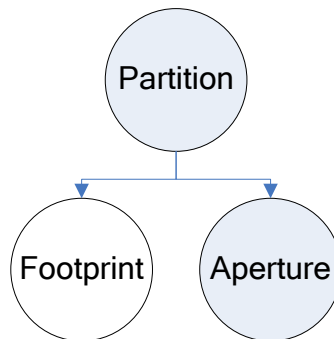
The <Room> tag is mandatory and is used to identify the two Rooms adjacent to the Partition<sup>59</sup>. For this reason, there must be exactly two <Room> tags. The path to an adjacent Room is as specified in section 6.5.5, Model Zone Naming. Note that UHRB defines the concept of an “outside” room when the partition defines a building outside wall. In CDB, the path of this outside room is \Shell\Exterior as illustrated in 6.5.6.4.1, Model Pseudo-Interior.

The <Surface> tag appears as many times as necessary to refer to all Surfaces making up this Partition. A path similar to the one used to refer to a Room is used to refer to a Surface. Note that a Partition does not refer to the Surfaces that belong to its Aperture; that will be taken care of by the Apertures themselves.

<sup>59</sup> Note that the CDB specification follows established UHRB conventions as it relates to partitions, namely that all partitions must be clipped so that there are no more than two neighbouring rooms.



The subgraph representing a Partition has the following structure.



**Figure 6-30: Partition Zone Structure**

The Footprint is the smallest polygon containing all of the referenced Surfaces when projected onto the XY plane. A Partition can have zero or more Apertures in it.

#### 6.5.6.4.7 Aperture Zone

An Aperture zone provides a mean by which one can enter or exit a Room. The Aperture zone is defined by the following XML tags.

**Table 6-16: Aperture Zone XML Tags**

```

<CDB:Zone name = "Aperture">
  <Label> aperture name </Label>
  <Is_Open> true/false </Is_Open>
  <Is_Fixed> true/false </Is_Fixed>
  <Damage_Level> percentage of damage </Damage_Level>
  <Room> path to room 1 </Room>
  <Room> path to room 2 </Room>
  <Surface> path to surface 1 </Surface>
  ... other surfaces as needed
</CDB:Zone>
  
```

The <Label> is optional. It allows the modeler to better identify the type of Aperture: door, window, trap, etc.

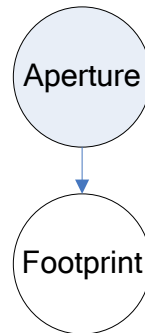
The <Is\_Open> and <Is\_Fixed> flags are both optional; they are considered false when not provided.

The <Damage\_Level> tag is also optional and provides a mean to indicate the level of damage of the Aperture. The value is expressed as a percentage using an integer in the range 0 (no damage) to 100 (destroyed).

The <Room> tag appears exactly two times and points to the two Rooms connected by this Aperture. Sometimes one of these two rooms may be an “outside” room - a concept defined in UHRB. In CDB, the path of this outside room is \Shell\Exterior as illustrated in 6.5.6.4.1, Model Pseudo-Interior.

The <Surface> tag appears as many times as necessary to refer to all Surfaces making up this Aperture.

The subgraph representing an Aperture has the following structure.

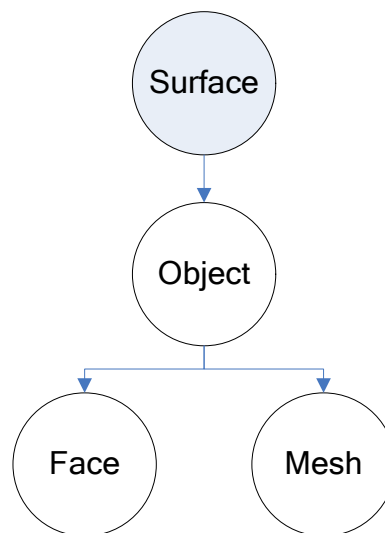


**Figure 6-31: Aperture Structure**

The Footprint is the smallest polygon containing all of the referenced surfaces when projected onto the XY plane.

#### 6.5.6.4.8 Surface Zone

A Surface zone contains useful geometry. That's all it does. The Surface zone is a plain CDB zone. Its subgraph is presented here.



**Figure 6-32: Surface Zone Structure**

A Surface is composed of one or more OpenFlight Object nodes holding the geometry defining the surface: face or mesh records.

## 6.6 Model Points

A model point is similar to a model zone; it identifies a location on the model that is of interest to at least one simulation client device. A point defines a local coordinate system on the model. Hence, a point has a position and an orientation.

In some respect, a point and a zone are similar and can be used interchangeably. A zone is used when the component of interest is physically modeled and has a graphical representation. When the zone is not modeled but still represents a component of interest, a point is used to indicate its presence.

Again, the OpenFlight Group Node mechanism provides a convenient means of implementing the concept of a point because a transformation can be added to the node.

### 6.6.1 Definition

The table below presents the syntax of the XML tags stored in the node's comment record.

**Table 6-17: XML Tags for Points**

```
<CDB:Point name = "name">  
... point attributes  
</CDB:Point>
```

The point name is mandatory while the point attributes are optional. In general, a point can have the same name as a zone. The following table lists the OpenFlight records required to represent a point.

**Table 6-18: OpenFlight Records for a Point**

```
GROUP  
MATRIX (mandatory)  
COMMENT (mandatory)
```

A model point is used in several occasions such as defining the attach point where another Model can anchor itself.

### 6.6.2 Usages

#### 6.6.2.1 Model DIS Origin

A Model that is intended as a DIS entity requires a point that defines the origin of the entity's coordinate system. This point is the center of the entity's bounding volume excluding its articulated and attached parts<sup>60</sup>. On a DIS network, the location of an entity is expressed relative to this point. There must be a single definition of this point for all damage states and all levels of details for a given model.

The following XML tag identifies the point.

<sup>60</sup> This definition can be found on page 3 of reference [4].

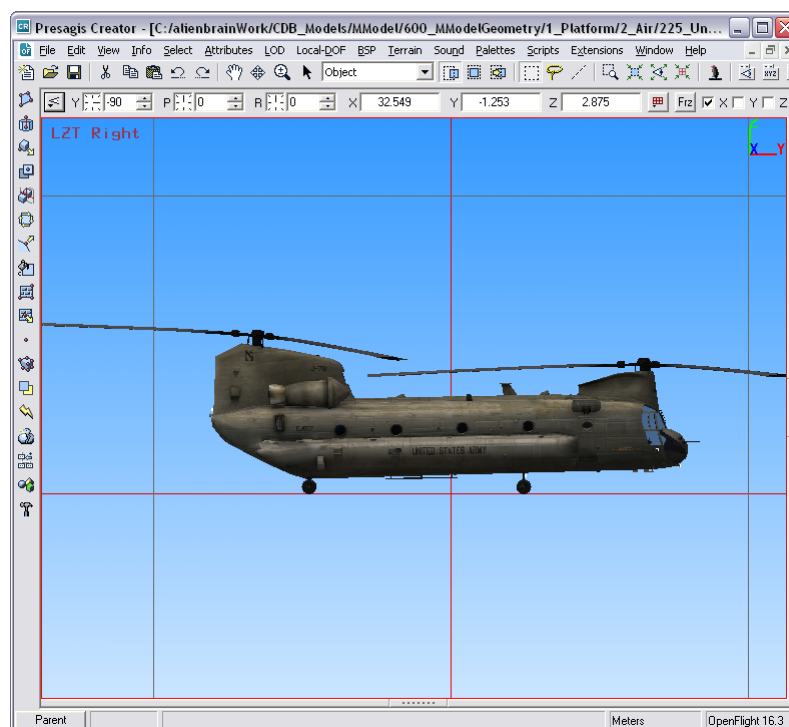
**Table 6-19: XML Tags for the DIS Origin**

```
<CDB:Point
  name = "DIS_Origin"
/>
```

If the DIS origin is not defined, it defaults to the model origin.

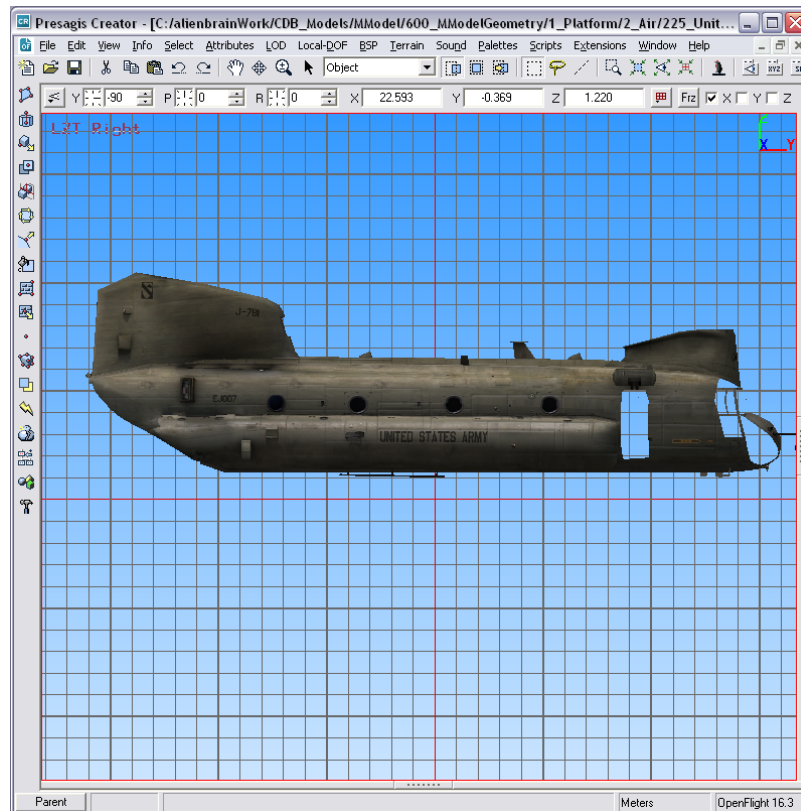
The CDB Point representing the DIS Origin must be positioned and oriented according the definition provided by the DIS Standard. This definition says that the DIS Origin is at the center of the bounding box of the entity, without articulated and attached parts. The standard also says what the orientation must be. The X-axis points forward, the Y-axis points to the right, and the Z-axis points down. All axes are aligned with the bounding box defined above.

The intent of the DIS Standard is to have its axis system aligned with the body of the entity. When it comes to air platform, the body is associated with the fuselage of the entity. To illustrate the difference in orientation between the DIS entity's bounding box and the CDB Model's bounding box, consider the Chinook helicopter shown below.



**Figure 6-33: Orientation of the Chinook Helicopter**

The fuselage of this helicopter has a pitch angle of approximately 1.6 degrees when resting on its wheels. Below is a snapshot of its fuselage, without rotors and landing wheels.



**Figure 6-34: The Body of the Chinook Helicopter**

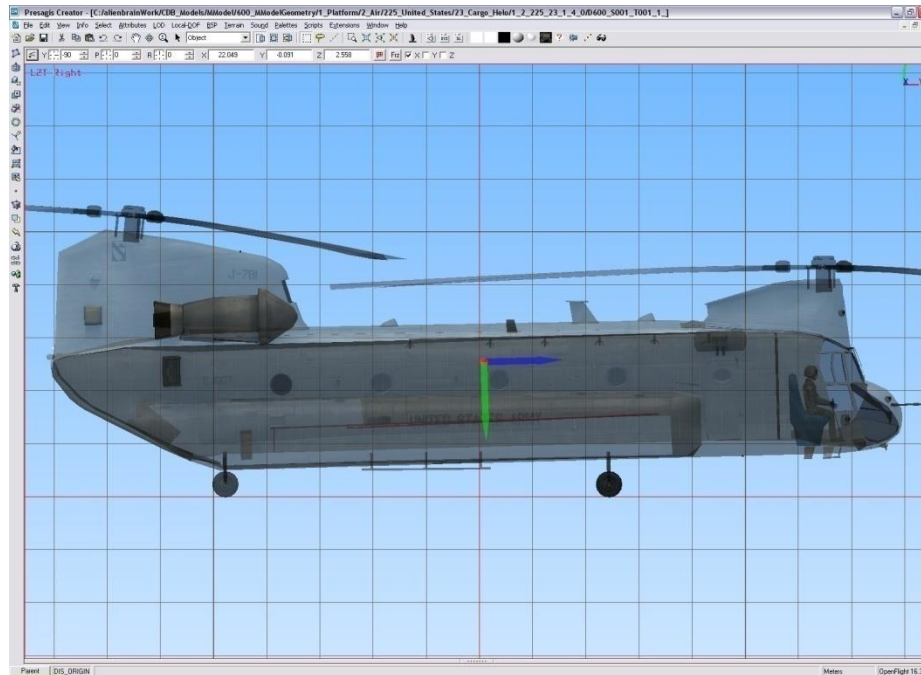
From the snapshots above, it is clear that the orientation of the DIS origin must be such that its XY plane makes an angle of 1.6 degrees with respect to the XY plane of the CDB axis system.

Here is a recommended way of defining the DIS Origin:

1. Create the Group Node and tag it as a CDB Point whose name is DIS\_Origin.
2. Make the CDB Point a child of the zone that best represents the entity's bounding volume without any articulated and attached parts.
3. Ensure the zone is properly oriented with respect to the CDB axis system.
4. Add a translation to position the origin at the center of the above bounding volume.
5. Add a rotation to align the X-axis to the front of this bounding volume.
6. Add another rotation to align the Z-axis with the bottom of the bounding volume.
7. By doing so, the Y-axis should already point correctly to the right side of the box.

### 6.6.2.1.1 Example

The snapshot below shows the proper location and orientation of the DIS origin on the Chinook. The DIS origin is represented by a set of 3 orthogonal Blue-Red-Green arrows. The blue arrow indicates the X axis; the green arrow points down and represents the Z axis.



**Figure 6-35: The DIS Origin of the Chinook Helicopter**

If you watch carefully, you will notice that the DIS axis system is aligned with the fuselage and makes an angle with the CDB XY plane.

### 6.6.2.2 Model Viewpoint

To generate the correct view of the outside world from a model's viewpoint, a client device needs an indication of where is the viewpoint located with respect to the model's origin. The viewpoint corresponds to the pilot's seat in an aircraft, the driver's seat in a ground vehicle, the navigation post on a ship bridge, the periscope on a submarine, or the eyes of a soldier. The viewpoint's local coordinate system is oriented such that the Y-axis indicates the viewing direction and the Z-axis points up.

The viewpoint has optional attributes to define the field of view available from this position. The field of view is defined by a frustum aligned along the local Y-axis. The horizontal field of view lies in the local XY plane while the vertical field of view is in the YZ plane.

**Table 6-20: XML Tags for a Viewpoint**

```
<CDB:Point name="Viewpoint">
  <FOV>
    <Horizontal>min max</Horizontal>
    <Vertical>min max</Vertical>
  </FOV>
</CDB:Point>
```

All values are expressed in degrees using decimal numbers. The default values are  $\pm 30.0^\circ$  in both directions for a total of  $60.0^\circ$  of horizontal and vertical fields of view.

### 6.6.2.3 Model Attach Point

A Model can be attached to another Model by mean of an attach point.

An attach point defines the position to which other (subordinate) models can attach themselves. For instance, a fighter has a number of attach points defined to receive missiles or external fuel tanks.

**Table 6-21: XML Tags for Attach Point**

```
<CDB:Point
  name = "Attach_Point"
/>
```

The orientation of the attach point is used to indicate how the two models connect together. A connection occurs by superimposing the coordinate system of the subordinate model with the coordinate of the attach point.

### 6.6.2.4 Model Anchor Point

The anchor point defines the location where a subordinate Model attaches to a parent Model. The anchor point is the counterpart to the attach point. Both can be seen as the male/female part of a connector and its receptacle.

**Table 6-22: XML Tags for Anchor Point**

```
<CDB:Point
  name = "Anchor_Point"
/>
```

The orientation of the anchor point is used to indicate how the subordinate model connects to the parent model. A connection occurs by superimposing the anchor point (of the subordinate model) with the attach point (of the parent model).

The default anchor point of a subordinate model is its origin.

### 6.6.2.5 Model Center of Mass

The Center of Mass (CM) of a Model is a specific point where, for many purposes, the Model behaves as if its mass was concentrated there.



Table 6-23: XML Tags for Center of Mass

```
<CDB:Point
  name = "Center_Of_Mass"
/>
```

## 6.7 Model Conforming

Historically, the integration of models onto the terrain has been performed during the database compilation process. These offline approaches varied considerably from vendor to vendor because there were no standardized approaches related to terrain meshing structures, varying visual priority and hidden-surface removal mechanisms, runtime LOD mechanisms, number of LODs, etc.

This section describes a series of model conformal modes that instruct client-devices on how they should conform models to the underlying terrain.

All of the conformal modes rely on the conforming of the model origin and/or model vertices onto the terrain mesh directly beneath the model. Note that the Z-component of the model's vertices is with respect to model's XY plane (as shown in Figure 6-36).

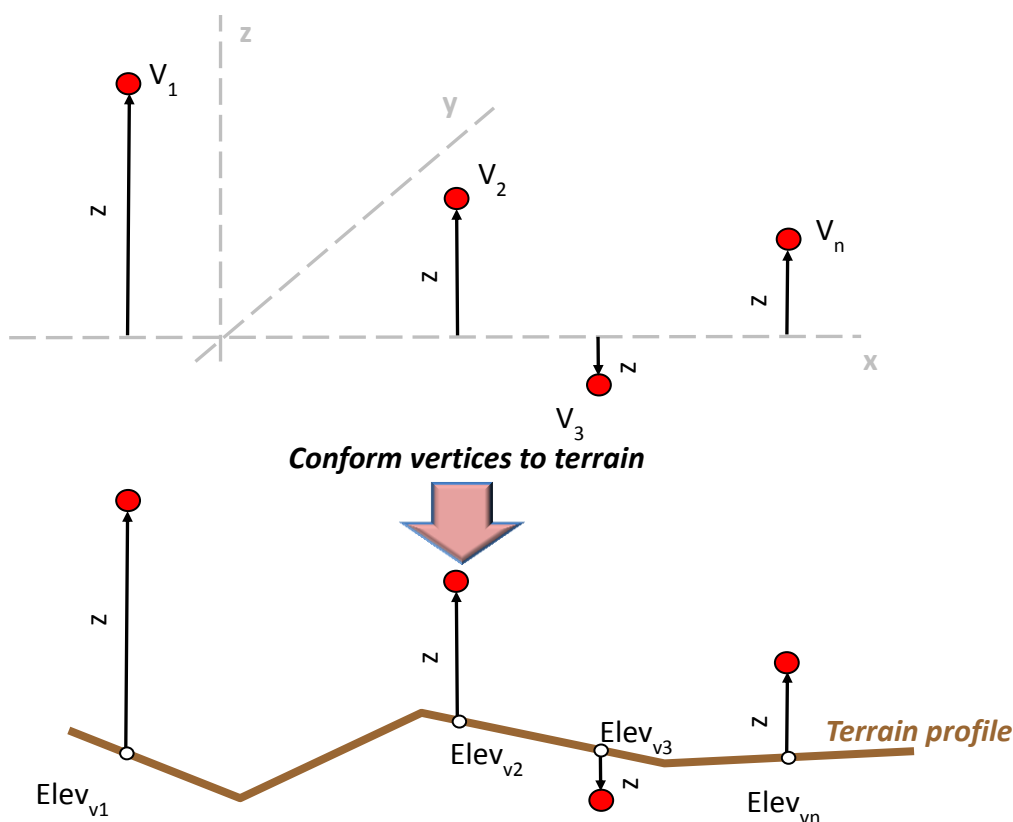


Figure 6-36: Conforming Vertices to Terrain

Note that by definition, all portions of the model below its XY plane represent some form of under ground basement. The conforming of models on steep or rough terrain may yield unusual results because portions of the basement may be visible. This may require the modelers to level the terrain in the immediate vicinity or adjust the model.

A modeler can specify a model's conformal mode by adding the following XML tags to the zone representing the model.

```
<CDB:Zone>  
  <Conformal mode="..." />  
</CDB:Zone>
```

The conformal modes are listed in Table 6-24 below.

**Table 6-24: Conformal Modes**

Conformal Mode
Absolute
Point
Vertex
Line
Plane
Surface

### 6.7.1 Non Conformal (Absolute) Mode

When attributed as a Non Conformal model, none of the model vertices are conformed to the underlying terrain. Instead the model's Z-values are used as-is, as elevation values. As a result the model is absolutely positioned and behaves independently of the terrain. The shape and orientation of the model is preserved. This conformal mode is typically used for the modeled representation of point-features. Typical use-cases include buildings, trees, and poles.

### 6.7.2 Point Conformal Mode

The Point Conformal mode conforms a single point of the model (its origin) onto the underlying terrain. All of the other model vertices are translated along the Z-axis; as a result, the shape of the model is preserved by this conformal operation. In effect, the Point Conformal mode dynamically positions a model on the underlying terrain so as to preserve the model's relative altitude over the terrain. Point-conforming is the default conformal mode for the modeled representation of point-features. Typical use-cases include buildings, trees, and poles.

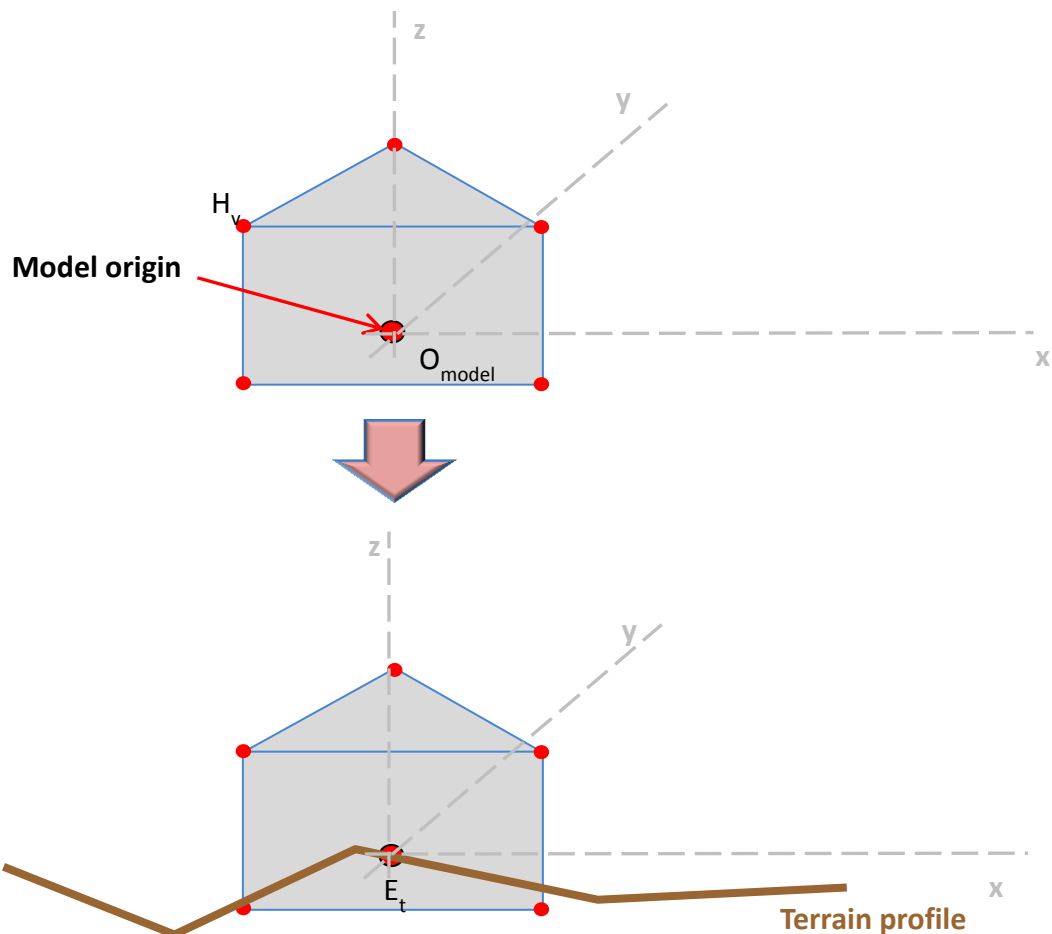


Figure 6-37: Origin Conformal Mode

### 6.7.3 Vertex Conformal Mode

The Vertex Conformal mode conforms each of the vertices of a model on the underlying terrain. The shape of the model is **not** preserved by this conformal operation. The model's XY plane defines a reference plane used by client-devices to adjust the elevation of each of the model's vertices. This conformal mode is used for 3D models that represent typically long 3D lineal features or large 3D areal features that need to follow the terrain profile. Typical uses include fences, walls, trenches, and forest canopies.

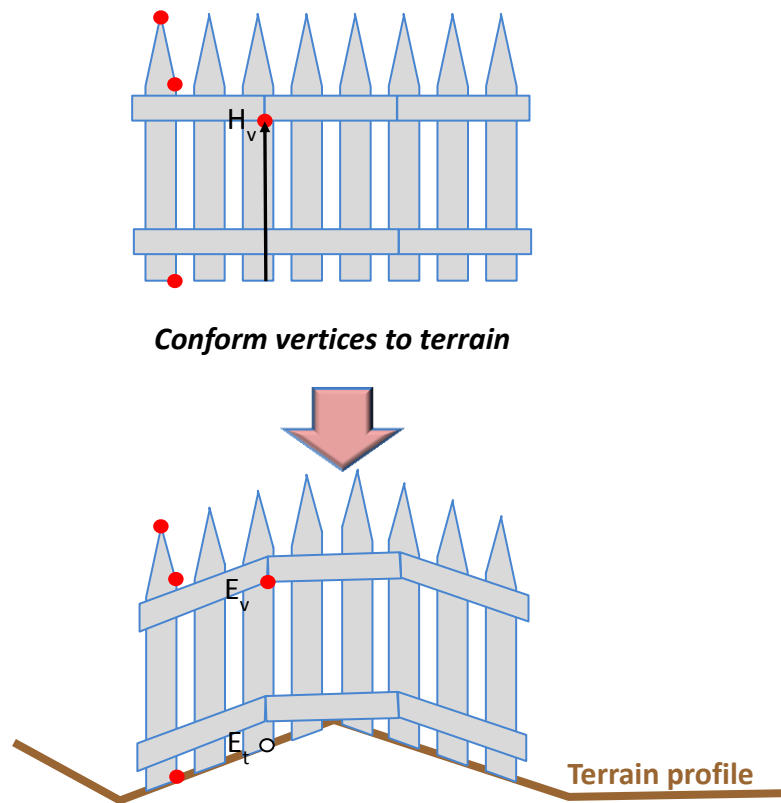


Figure 6-38: Vertex Conformal Mode Example

#### 6.7.4 Line Conformal Mode

The Line Conformal mode conforms each of the two reference vertices of the “linear” model on the underlying terrain. All of the other model vertices are sheared along this axis; as a result, the shape of the model is **not** preserved by this conformal operation. The model’s XY plane defines a reference plane used by client-devices to adjust the elevation of the two reference vertices. This conformal mode is used for models that represent linear features such as powerlines and monorails.

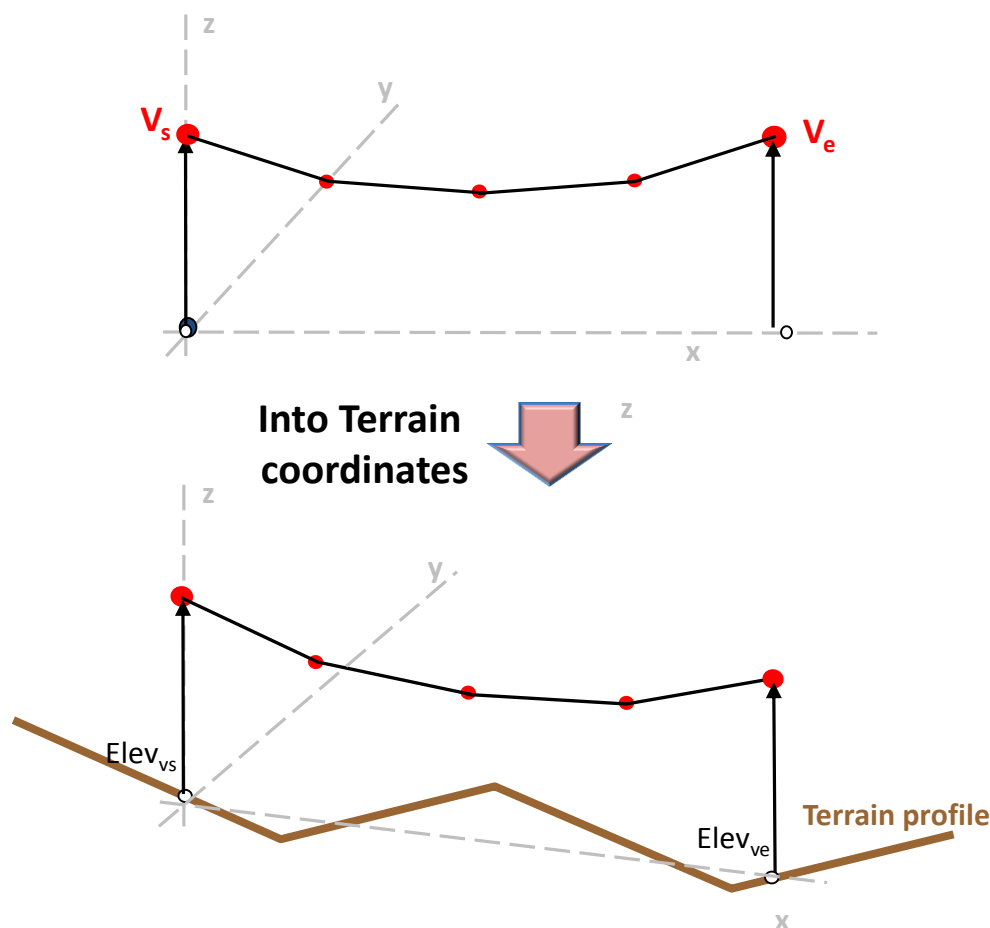


Figure 6-39: Line Conformal Mode

The line that is used to specify the conforming is defined by a Face node with the following XML tags:

```
<CDB:Face>
  <Conformal_Line/>
</CDB:Face>
```

This Face node defines a single line with two vertices, the first one,  $V_s$ , being the start and the second,  $V_e$ , the end of the line.

### 6.7.5 Plane Conformal Mode

The Plane Conformal mode conforms each of the three reference vertices of the “planar” model on the underlying terrain. The resulting three vertices define a model transformation matrix that can then be applied to the vertices of the model. As a result, the shape of the model is preserved by this conformal operation, but the model

undergoes a change in pitch and roll angles. Given this property, there are relatively few cases where this conformal mode can be used<sup>61</sup>. However, as shown in Figure 6-41, this conformal mode is required when conforming the curved segments of 3D (raised profiled) modeled road features.

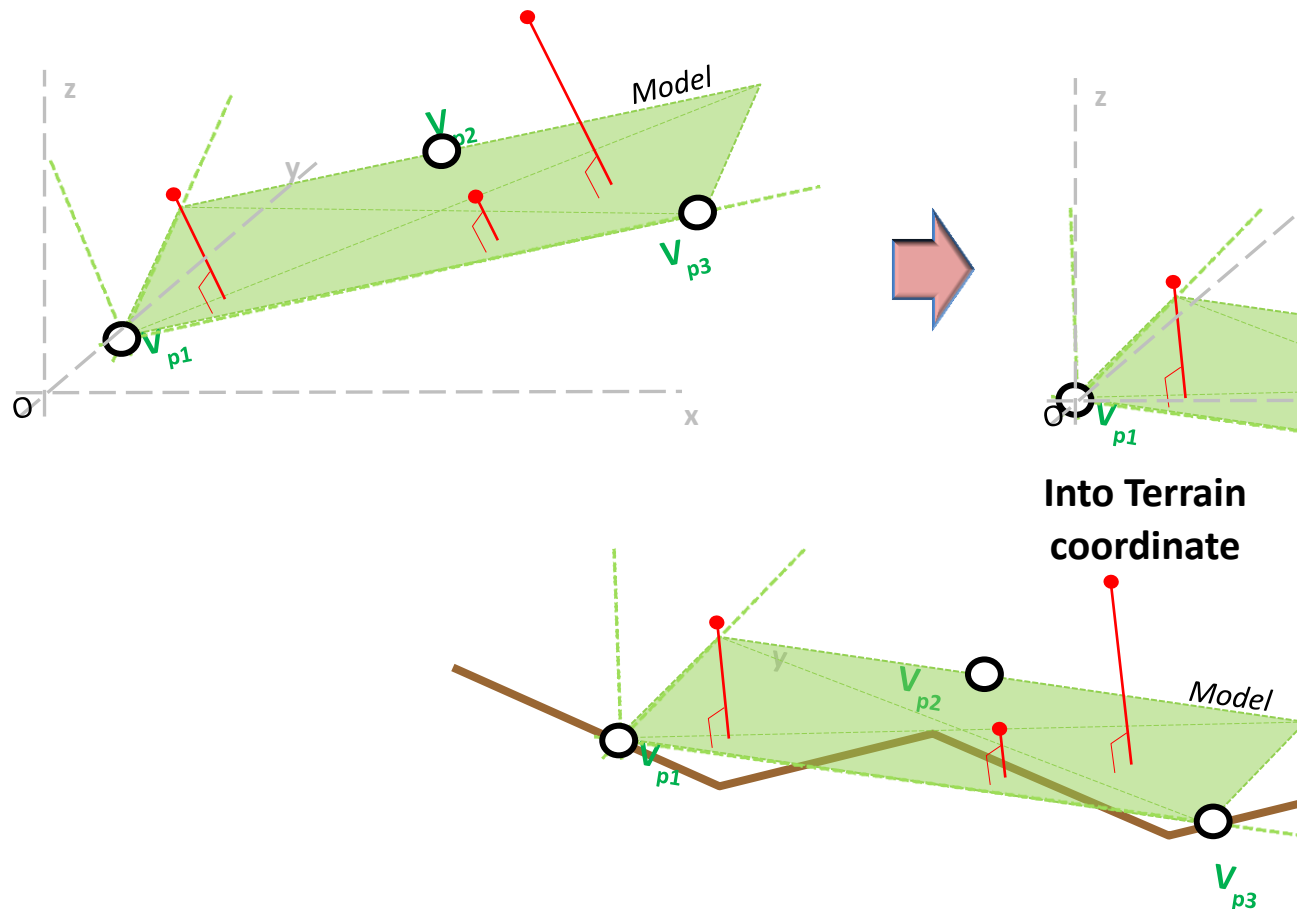
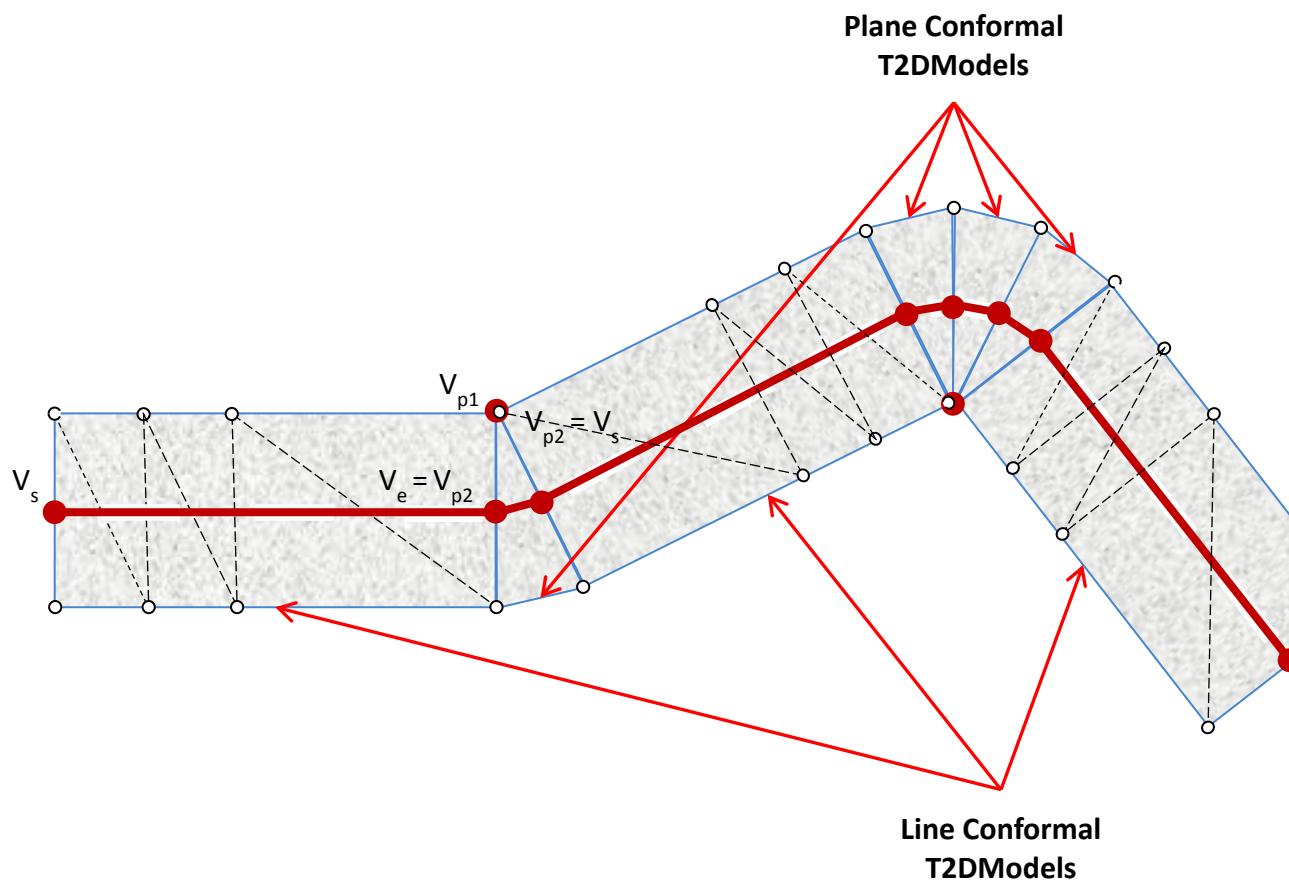


Figure 6-40: Plane Conformal Mode

<sup>61</sup> Man-made structures and tree vegetation do not tilt regardless of the terrain they are on.



**Figure 6-41: Application of Line and Plane Conformal Modes on 3D Roads**

The plane that is used to specify the conforming is defined by a Face node with the following XML tags:

```
<CDB:Face>
  <Conformal_Face/>
</CDB:Face>
```

The Face node has exactly 3 vertices defining the plane used for the conforming. The only restriction on these 3 vertices is that they must not be collinear.

### 6.7.6 Surface Conformal Mode

This conformal mode is used for models whose points, edges and surfaces must all conform exactly to the underlying terrain. The Surface Conformal mode requires that the model's edges and surfaces be clipped to the underlying terrain. The original vertices and the added vertices resulting from the clipping operation are conformed to the underlying terrain. As a result, the shape of the model is **not** preserved by this



conformal operation. This conformal mode is primarily used for the modeled representation of 2D surface-feature such as paint markings and other terrain overlays. In addition, it can be used for 3D models that represent typically long or large 3D lineal and 3D areal features that need to **perfectly** follow the terrain profile. Note that in most cases, the vertex conformal mode provides an adequate solution for 3D models and is more economical to use than the surface conformal mode.

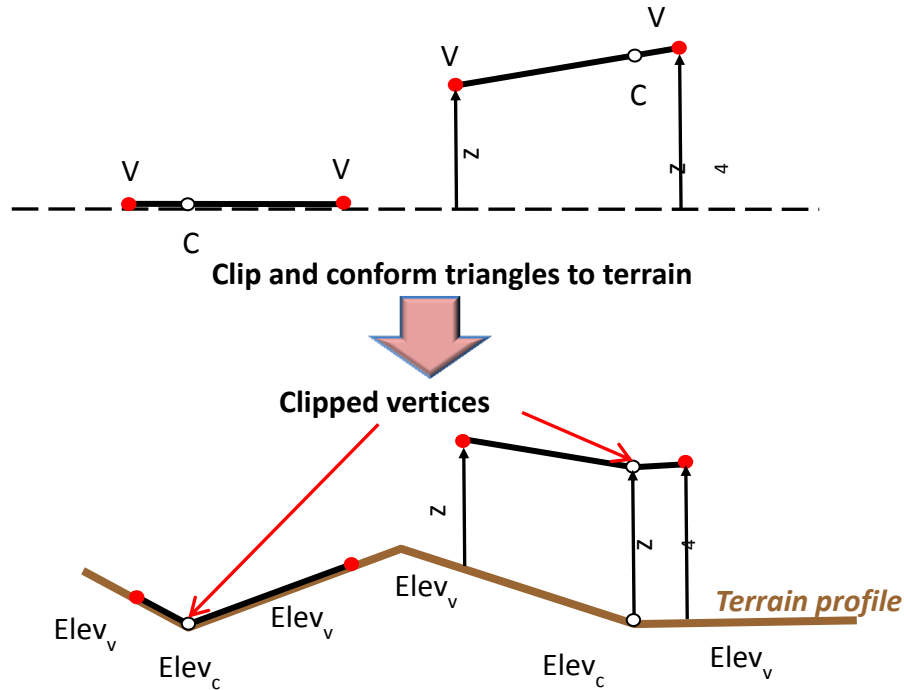


Figure 6-42: Surface Conformal Mode

## 6.8 Model Levels-of-Detail

A levels-of-detail model structure is essential when the intent is to use a model in a real-time application such as flight simulation. The level-of-detail mechanism provides client-devices with the essential structure for deterministic operation. Deterministic operation can be achieved only if a client-device can:

- control the paging bandwidth from the CDB main storage device
- control client-device processing load
- control client-device memory footprint
- control run-time publishing processing load and
- control run-time publishing memory footprint

For this reason, it is recommended to create LODs, especially for complex Models, and for models that are used extensively, in great density in the CDB. This is most critical for geospecific cultural models (especially in densely modeled geospecific areas of the synthetic environment) since they can consume a significant portion of



the paging bandwidth and memory footprint of the client-devices. As a corollary, simple Models should not be made more complex by adding unnecessary level of details. The CDB Specification provides rules for determining model complexity, and selecting the appropriate LOD, as defined in this section.

OpenFlight LOD nodes now support two methods of specifying the criteria to determine if a level of detail is active, that is if the user application should traverse the node and its children. The first method, the classic one, is to specify the switch in and switch out distances in real world units. Using this method, a level of detail is active when the distance from the viewpoint to the center of the LOD is within the switch-in and switch-out distances. The second method uses the Significant Size associated with the LOD node to determine when to activate the node. **The CDB Specification requires using the second method as it makes the subject (i.e., the model) independent from its observers (i.e., the client-devices).**

There are several problems associated with the classic, range-based method. In a visual system for instance, the switching distance should be based on both range and the system resolution of the entire visual system; a database designed to rely solely on a range-based switching criteria is not truly portable, especially if the intent is to use it on systems with wildly different visual resolution. Furthermore, the blending or morphing of models solely based on range criteria can lead to undesirable effects. When the viewpoint moves quickly, the distance over which the model is LOD-transitioning should be large enough to avoid the “popping-in” of the higher LOD version. On the other hand, if the viewpoint is moving very slowly, the distance over which the model is LOD-transitioning should be reduced to avoid the “LOD-ghosting” of the higher LOD version. These two constraints make implicit assumptions on the model’s speed. In applications where the aircraft’s flight regime varies considerably (V22 for example), it is impossible to find a single set of LOD start and end points that simultaneously cater to all flight modes (hover versus cruise). Here again, a database design that directly encodes the start and end points of a model’s LOD transition is not truly portable, because it makes implicit assumptions on the speed it will be used for. Thus, in a tactical fighter application, the start and end points of a model’s LOD transition need to be widely spaced apart to prevent a popping effect at the onset of the LOD transition. Conversely, in a tank application, the start and end points of a model’s LOD transition need to be much more closely spaced to prevent a ghosting effect as the higher LOD model is blended-in. If the client device wants to implement some form of transition between LODs, the criteria should be based on a user-defined duration. Transitions between LODs can involve fading in the next LOD while fading out the current one. That fading operation should not last forever. It should be accomplished in a relative short period of time. The second method to transition from one LOD to the next is to use morphing. In the case of morphing, the transition period is less critical because the client-device (typically an Image Generator) does not blend-in two models together.

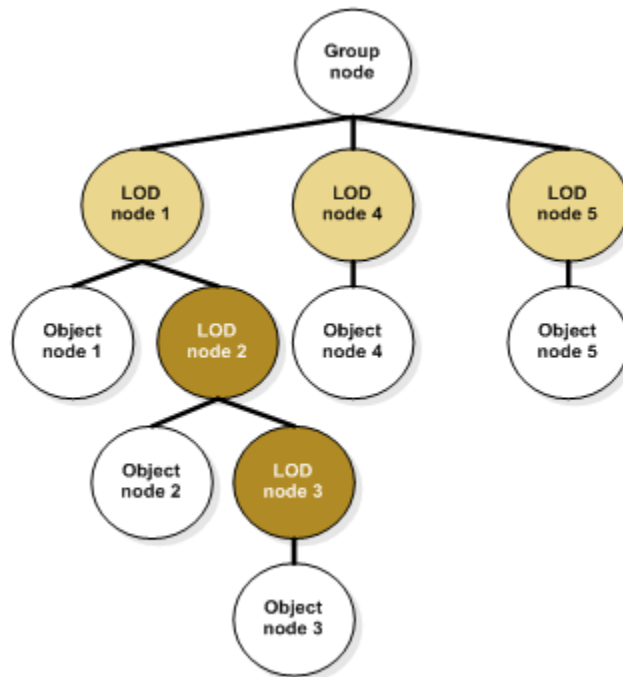
The consequences of such implicit assumptions result in a database that is highly client-device, and application-specific.

For all these reasons, the CDB Specification has selected the second method to control the LOD mechanism.

### 6.8.1 LOD Node Types

Two methods exist to implement LODs, addition or exchange. The two methods can be used simultaneously and are not mutually exclusive.

In the first method, details are progressively added to the model, as the viewpoint gets closer. With the second method, different representations of the same model are substituted for one another. Figure 6-43: Exchange and Additive LOD Nodes, illustrates the general organization of Models with both types of LOD nodes.



**Figure 6-43: Exchange and Additive LOD Nodes**

The LOD nodes 1, 4, and 5 represent Exchange LODs and are mutually exclusive. LOD nodes 2 and 3 are considered Additive LODs because they do not have other immediate sibling nodes of type LOD.

#### 6.8.1.1 Note on Additive LODs

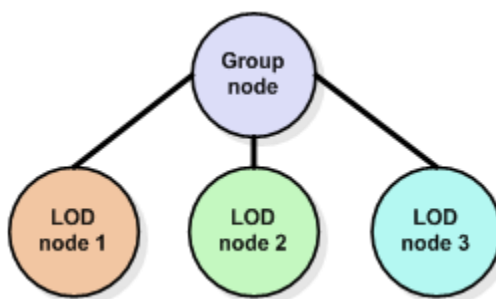
An Additive LOD is just a special case of the more general Exchange LOD paradigm. When a LOD node has no sibling LOD, it becomes an Additive LOD node. That does not change the fact that at most one LOD node gets selected based on its Significant Size.

## 6.8.2 LOD Node Ordering

The OpenFlight Specification does not impose any constraint on the ordering of sibling LOD nodes. For this reason, all nodes would have to be tested because the runtime system is searching for the LOD node with the smallest Significant Size that still contributes to the resulting image.

Consequently, the CDB Specification requires that LOD nodes be ordered. This ordering improves client device performance without being specific to any client-device (since it can be costly to test all nodes to select just one).

To illustrate the need for ordering LOD nodes, consider the case where a modeler needs to create a realistic representation of a small town with several hundred buildings. The level of details of the town must accommodate low altitude flight with a helicopter. After a few tests, the modeler decides to model each building with three levels of details as shown in Figure 6-44: Exchange LOD Nodes.



**Figure 6-44: Exchange LOD Nodes**

Now, consider the case where the simulated ownership has an entire city visible (thousands of Models) within the field of view of the visual system. This situation forces the IG client device to test all of the LOD nodes of all Models if these nodes are not sorted. However, if the nodes are sorted, it is possible to test only a subset of all nodes to find out which ones to display. Which order is best? Ascending or descending order?

If all buildings are visible, the majority will be located far from the viewpoint while only a few will be near the viewpoint. In general, only few models fit near the viewpoint, and that number increases with the distance.

If LOD nodes are sorted in ascending order of their Significant Size, the client device quickly selects the finest LOD of Models located near the viewpoint. Indeed, only one test is necessary to select the correct LOD node in these models. However, for Models located farther, the client device has to perform two or three tests to select the correct nodes. If LOD nodes are sorted in ascending order, a single test is done on a small number of LOD nodes while two or three tests are performed on the majority.

On the other hand, if nodes are sorted in decreasing order of their Significant Size, the client device performs three tests to select the finest LOD only on a limited number of models near the viewpoint. For a larger number of models, two tests are required.

Moreover, for an even larger number of models, the ones located far away from the viewpoint, a single test is enough to select the coarsest LOD, if it gets selected at all.

The second approach allows for a smaller number of tests to select the correct LODs, and is the method selected by the CDB Specification. **As a result, LOD nodes must be sorted in decreasing order of their Significant Size attribute. As a corollary, the CDB Specification also requires that sibling LOD nodes be mutually exclusive.**

### 6.8.3 LOD Significant Size

The concept of a Significant Size is a recent improvement of the OpenFlight Specification. When a finer model LOD is created, the modeler typically adds additional geometric detail, additional features (such as markings), or refines the shape of curved surfaces (such as engines, wheels), etc. When assigning a Significant Size to a model LOD, the modeler needs to answer the following question: When I created a new model LOD, I did so to create additional detail in my model. What is the largest dimensional change in geometry for this new model LOD? In other words, what is the largest dimensional difference between this LOD and its previous and coarser LOD? In effect, the value of Significant Size corresponds to the “modeling difference” between the LOD and its previous coarser LOD. At runtime, a client-device converts this modeling difference value from its real-world dimensional value into a viewing error value (typically measured in pixels or degrees). The client-device can then select the appropriate model LOD because it knows that the modeler’s intent in creating the LOD was to show features, eliminate all modeling discrepancies whose dimension equaled that of the Significant Size dimension associated with that model’s LOD. This contribution of the LOD to the scene is based on the LOD’s Significant Size as well as other parameters (such as system resolution) relevant to the simulation model used by the client device.



Version 16.0 of the OpenFlight Specification introduces the concept of Visual Significance that is different from the concept of Significant Size. The concept of Visual Significance translates in two fields called Significance and they are found in the Group Record and Object Records. Here is the definition of this field as found in reference [11]:

*“Significance can be used to assist real-time culling and load balancing mechanisms, by defining the visual significance of this group with respect to other groups in the database. Normally the value of this attribute is zero”.*

The CDB Specification mandates a value of zero for Visual Significance; the value zero indicates the object or the group has no particular significance and is not more or less important than any other objects or groups. Any other values, whether negative or positive, are reserved for future use by this Specification.

### 6.8.3.1 Definition of Significant Size

The Significant Size is defined as the “size” of the model, expressed in meters. By extension, it applies equally well to a submodel represented by an Additive LOD. In the case of an Exchange LOD, the Significant Size is the difference between two representations of the model or submodel.

### 6.8.3.2 Estimating the Size of the Model

Many models have shapes that resemble a cube (with roughly equal length, width, and height), and thus their significant size can be simply estimated by the length of the diagonal of their bounding box. As the shape of a model departs from that of a simple cube, either with respect to aspect ratio, or with respect to the amount of negative space within its bounding box, the model’s significant size should be decreased proportional to the amount of departure.

### 6.8.3.3 How to use the Significant Size

The Significant Size is used to distribute models into appropriate CDB LODs. Once a value is assigned to the coarsest LOD of a model, subsequent LODs of the same model can be distributed to subsequent CDB LODs. Use Table 3-1 and the model Significant Size to identify the CDB LOD it belongs to.

For instance, if the size of a building is estimated to 75 meters, then its coarsest LOD will be stored in CDB LOD 0, according to Table 3-1. On the other hand, a 2-meter park bench will appear in CDB LOD 5.

### 6.8.4 LOD Limits

The number of vertices per LOD is limited to ensure a smooth progression between all representations of the same model. Table 6-25 below gives the maximum number of vertices allowed for each Model-LOD.

**Table 6-25: Maximum Number of Vertices per Model-LOD**

Model-LOD	Maximum Number of Vertices	CDB LOD
0	128	CDB LOD <sub>c</sub> + 0
1	512	CDB LOD <sub>c</sub> + 1
2	2048	CDB LOD <sub>c</sub> + 2
3	8192	CDB LOD <sub>c</sub> + 3
4	32768	CDB LOD <sub>c</sub> + 4
5	131072	CDB LOD <sub>c</sub> + 5
6	524288	CDB LOD <sub>c</sub> + 6
7	2097152	CDB LOD <sub>c</sub> + 7

The table above shows that a model is allowed up to 8 levels of details, numbered from 0 to 7. Model-LOD 0 is the coarsest level of detail of the model and may count up to 128 vertices. As the complexity of subsequent Model-LODs augments, a higher vertex count is permitted. The CDB LOD that is associated with a Model-LOD is expressed relative to the CDB LOD assigned to its coarsest representation, designated by CDB LOD<sub>c</sub>.

#### 6.8.4.1 How to Assign CDB LODs

To illustrate the use of Table 6-25, take a 3D model representing a building with three representations:

- Coarsest LOD:
  - 28 vertices
  - Significant Size estimated to 25 meters
- Medium LOD:
  - 2244 vertices
  - Significant Size estimated to 4 meters
- Finest LOD:
  - 10320 vertices
  - Significant Size estimated to 10 cm

The CDB LODs are first established by looking up the Significant Sizes of the three representations in Table 3-1:

- Coarsest LOD:
  - 25 m is CDB LOD 2
  - This is CDB LOD<sub>c</sub>
- Medium LOD:
  - 4 m is CDB LOD 4
- Finest LOD:
  - 10 cm is CDB LOD 10

We then use the vertex count to identify the Model-LOD in Table 6-25:

- Coarsest LOD:





- 28 vertices is Model-LOD 0
- Medium LOD:
  - 2244 vertices is Model LOD 3
  - Should be assigned to CDB LOD<sub>c</sub> (2) + 3 = 5
- Finest LOD:
  - 10320 vertices is Model LOD 4
  - Should be assigned to CDB LOD<sub>c</sub> (2) + 4 = 6

Since all representations of the model must meet both the constraints associated with the Significant Size and the Vertex Count, the final CDB LODs are the maximum ones identified above.

- Coarsest LOD:
  - CDB LOD 2
- Medium LOD:
  - CDB LOD 5
- Finest LOD:
  - CDB LOD 10

## 6.8.5 LOD Generation Guidelines

The following guidelines should help modelers produce efficient CDB models for use in real-time environments. There are two way of proceeding; one way is to create the finest representation of the model and then simplify it until the coarsest representation is obtained. The other way consists in creating the coarsest representation first and then refining it until the desired representation is obtained.

In general:

- The coarsest Model-LOD is the simplest possible geometric representation of the model using at most 128 vertices.
- A coarser Model-LOD is created by removing details from a finer Model-LOD.
- Alternately, a finer Model-LOD is created by adding details to a coarser Model-LOD.
- In both cases, the size of the details that are removed or added to a Model-LOD should be consistent with its Significant Size.
- Model-LOD 0 is mandatory; the others are optional and exist only if Model-LOD 0 isn't sufficient to represent the model with a proper level of detail.
- Multiple Model-LODs do not need to be consecutive.

## 6.9 Model Switch Nodes

A Switch Node allows the selection of zero or more children by invoking a selector mask. Any combination of children can be selected per masks and the number of definable masks is unlimited. The CDB Specification makes use of OpenFlight Switch Nodes to control the state of Model Components (zones and points).

## 6.9.1 Definition

XML tags in the comment record are added to the switch’s primary record to identify it as a CDB Switch.

**Table 6-26: XML Tags to Create a CDB Switch**

```
<CDB:Switch name = "switch name">
... switch attributes
</CDB:Switch>
```

The switch must contain one mask per state. As an example, if the switch has 3 children, each representing a separate state of the parent zone, then the switch needs 3 masks, each selecting one child.

In addition to defining a mask for each switch state, the CDB Specification demands that each mask be named. The name of the mask must be representative of the state selected by that mask. The actual name is at the discretion of the modeler.

The corresponding OpenFlight records are as follow:

**Table 6-27: OpenFlight Records to Create a CDB Switch**

```
SWITCH
COMMENT (mandatory)
INDEXED STRING
```

Note that the first mask, mask index 0, is the default mask. This means that the value of the Current Mask field in the Switch record must be 0.

## 6.9.2 Usage

### 6.9.2.1 Articulations with Discreet Positions

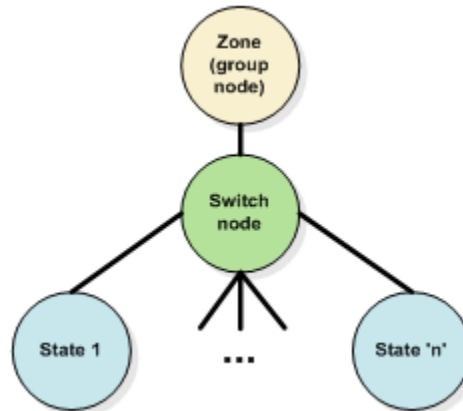
Switch nodes provide an alternative to DOF nodes when an articulated part is implemented for only a few positions. An example of this use of switches is the control of undercarriage or control surfaces on aircraft. Suppose the modeler wants to represent the flaps in two distinct positions: flaps up and flaps down. A switch is the simplest way to implement these two flaps positions. In this example, the switch name could be “Flap Control” and the two mask names could be “Flap Up” and “Flap Down”.

Suppose the modeler wants to provide two positions for the door on a hangar: open and close. In addition, when the door is open, the modeler provides a representation for the interior of the hangar, which is not the case when the door is closed. Again, the use of a switch is appropriate to provide the control over the door position. A proper name for the switch would be “Door Position” and the appropriate names for the two masks would be “Door Closed” and “Door Open”.

### 6.9.2.2 Damage States

Switch nodes can be used to select one of many modeled representations of damages. A zone has at least a normal (usually undamaged) state. When other states exist, an

OpenFlight switch node is used to select which state is active. A single damage state can be active at any time. Figure 6-45: General Damage State Tree Structure shows the general organization of a zone with several states.



**Figure 6-45: General Damage State Tree Structure**

Each damage state represents the zone with a certain level of damage. This level of damage is expressed as a percentage from 0 to 100%. A level of damage of 0 % means the zone is not damaged at all. At the opposite end, a percentage of damage of 100 % indicates the zone is completely destroyed.

To identify a damage state switch, use the following XML tags in the switch comment record.

**Table 6-28: XML Tags for Damage State Switch**

```
<CDB:Switch name = "Damage_State">
  <Damage_Level>...</Damage_Level>
</CDB:Switch>
```

The XML element `<Damage_Level>` is a list of percentages representing the transitions between child nodes of the switch. The list counts ‘n-1’ entries where ‘n’ is the number of states.

The percentages representing the transitions are limited to the range [0, 99]. The value 100 is not allowed because the level of damage must exceed the transition value in order to select the correct state.

To illustrate the concept of level of damage, assume a damage state switch has 3 child nodes representing the zone in normal, damaged, and destroyed states. Also, assume that the modeler’s intent is to switch to the damaged state when the level of damage exceeds 25 %, and to switch to the destroyed state when the level of damage exceeds 75 %. Here, the XML tag associated with the switch should look like this.

**Table 6-29: Example of a Damage State Switch with Two Transitions**

```
<CDB:Switch name = "Damage_State">
  <Damage_Level>25 75</Damage_Level>
</CDB:Switch>
```

The ordering of damage states must be from left (normal state) to right (destroyed state). All intermediate states must represent increasingly damaged states from a slightly damaged state to an almost destroyed state. The number of states is left to the discretion of the modeler.

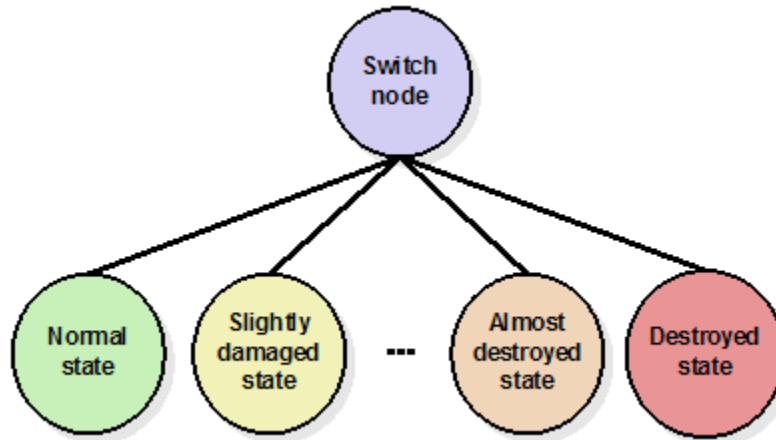


Figure 6-46: Damage States Ordering

While the number of damage states is left to the discretion of the modeler, some choices are better than others. Since a Model is meant to be used in a simulator and since many simulators are DIS-compliant, it is suggested to create the same number of CDB damage states as there are DIS damage states for the corresponding entity.

For instance, if the Model represents a DIS land platform such as the M1A2 tank, the modeler could create four damage states to match the four corresponding DIS damage states labeled No Damage, Slight Damage, Moderate Damage and Destroyed.

The DIS and the HLA standards are relatively vague regarding the definition of damage states. In the case of the DIS standard, the damage state is a field that belongs to a structure called the Entity Appearance. The field has only 2 bits and, accordingly, accommodates four different values. For HLA, version 2 of the RPR-FOM defines the damaged appearance as a 32-bit enumeration for which only 4 values have been defined so far – the same values as the one defined by DIS, that is No Damage, Slight Damage, Moderate Damage and Destroyed.

For both DIS and HLA, it is obvious that the damage state is meant to be a visual damage state.

The question to answer is the following: “What should the universally accepted visual appearance be for a slightly (or moderately) damaged state?”

In the DIS world, a platform is often qualified in terms of Mobility and Fire Power<sup>62</sup>. Using these two criteria, it is possible to define the following guidelines.

---

<sup>62</sup> Note that, on top of the Damage State field, the DIS Entity Appearance structure has two flags to describe the Mobility and Fire Power of the entity. This is also true for HLA and version 2 of the RPR-FOM which provides for two flags to describe the fire power and mobility of a physical entity on top of the field used to describe the damage state.

- A slightly damaged model should represent a platform with limited mobility. However, its firepower is intact and it should be apparent that the entity is still capable of firing its weapons.
- A moderately damaged model should represent a platform for which both mobility and firepower are reduced without being completely out of service.

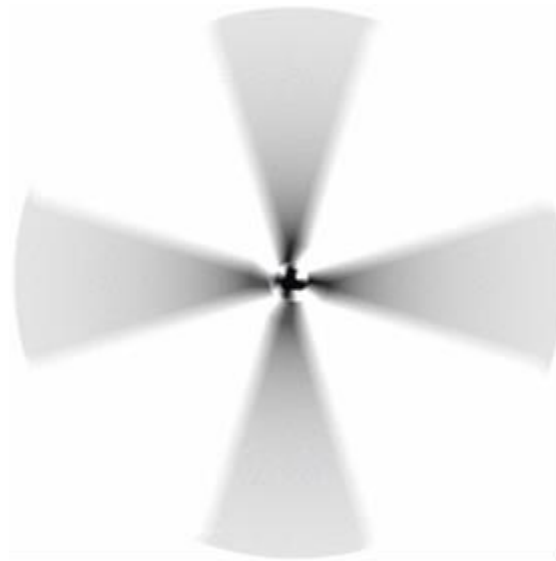
As a corollary, here are the definitions of normal and destroyed states.

- An undamaged model should represent a platform for which both mobility and fire power are completely operational.
- A destroyed model should represent a platform for which both mobility and firepower are completely out of service.

### **6.9.2.3 Temporal Anti-aliasing**

Temporal anti-aliasing may be achieved with the use of special textures. These textures are often required to aid IG client-devices to eliminate strobing effects on model rotating objects such as helicopter rotors, aircraft propellers, or vehicle wheels.

Figure 6-47: Example of a Texture Representing a Rotor, is an example of a semi-transparent texture used to simulate a rotating helicopter rotor.

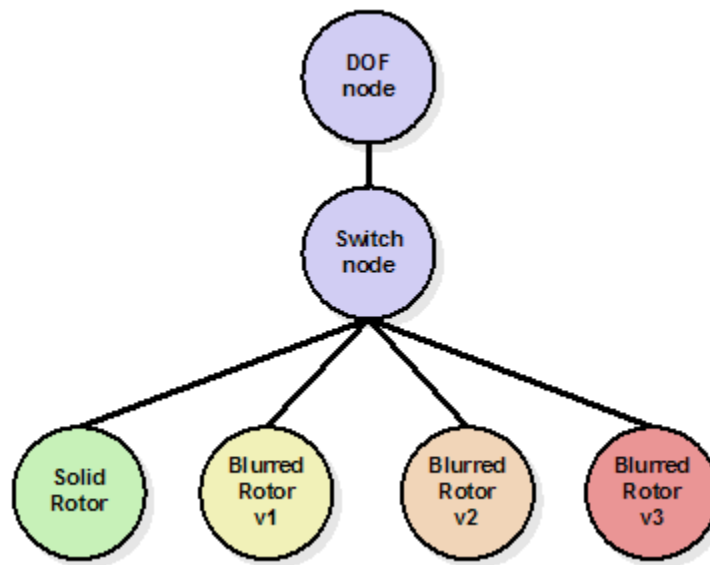


**Figure 6-47: Example of a Texture Representing a Rotor**

Motion blur textures are general base textures with a Texture Kind of S001. The Texture Index (Tnn) is used to sequentially number several motion blur textures representing the same object.

The use of motion blur textures can be combined with DOF and Switch nodes to produce efficient switching between several versions of a single rotating part.

The following subtree illustrates how four versions of the above rotor could be modeled using one solid version and three blurred versions.



**Figure 6-48: Multiple Versions of Rotating Parts**

In this example, three textures are used to represent an increasingly blurred rotor.

In order to detect the presence of the above construct, the following XML comment must be added to the switch node.

**Table 6-30: XML Tags for Motion Blur Switch**

```

<CDB:Switch name = "Motion_Blur">
  <Blurriness>...</Blurriness>
</CDB:Switch>
  
```

The children of the switch node could be any OpenFlight nodes. Most likely, the nodes that contain the geometry will be OpenFlight Object nodes.

When modeling solid and blurred objects in this manner, the CDB Specification requires that the leftmost child node contains the solid version of the object while the sibling nodes to the right contain increasingly blurred version of the same object.

The XML element `<Blurriness>` is a list of percentages representing the transitions between child nodes of the switch. The list counts ‘n-1’ entries where ‘n’ is the number of child nodes.

The percentages representing the transitions are limited to the range [0, 99]. The value 100 is not allowed because the level of blurriness must exceed the transition value in order to select the correct child node.

To illustrate the concept of level of blurriness, assume a motion blur switch has two child nodes. Also, assume that the modeler’s intent is to switch to the second node when the level of blurriness exceeds 10 %. Here, the XML tag associated with the switch should look like this.



**Table 6-31: Example of a Motion Blur Switch with One Transition**

```
<CDB:Switch name = "Motion_Blur">  
  <Blurriness>10</Blurriness>  
</CDB:Switch>
```

## 6.10 Model Articulations

### 6.10.1 Definition

An OpenFlight DOF node is used to implement the concept of a CDB Articulation. The node gives the modeler controls over all 9 degrees of freedom, translation, rotation and scaling on all 3 axes. Generally, only one degree of freedom is allowed at a time and most often, that single degree of freedom is a rotation about a single axis. However, the modeler is free to allow any translation, rotation and, even scaling; even though stretching an articulation does not usually produce a realistic effect.

Since only one articulation is allowed per zone, the zone name is sufficient to identify and control the DOF node.

A CDB Articulation node is an OpenFlight DOF node with attributes in the form of XML tags. The table below presents the syntax of the XML tags stored in the DOF node's comment record.

**Table 6-32: XML Tags for DOF**

```
<CDB:Articulation name="name" id="id">  
  
  <Translation>  
    <X rate="value" />  
    <Y rate="value" />  
    <Z rate="value" />  
  </Translation>  
  
  <Rotation>  
    <X rate="value" />  
    <Y rate="value" />  
    <Z rate="value" />  
  </Rotation>  
  
  <Scaling>  
    <X rate="value" />  
    <Y rate="value" />  
    <Z rate="value" />  
  </Scaling>  
  
</CDB:Articulation>
```

The above XML tag is necessary in two circumstances:

1. The articulation represents a DIS Articulated Part.
2. The articulation is to be animated automatically.



A CDB Articulation node has an optional name that is used to self-document the articulation. The optional identifier provides the corresponding DIS Articulated Part. It is suggested to use a name inspired from the DIS Articulated Part ID, when the identifier is supplied. For instance, DIS identifies as Primary Gun 1 the articulated part whose ID is 4416. That example would generate the following XML tags:

```
<CDB:Articulation name="Primary Gun 1" id="4416" />
```

Section 4.7.3 in reference [4] provides a list of DIS Articulated Part IDs.

It is possible to specify an optional Rate-of-Change for each Degree of Freedom along their X, Y, and Z axes for Translation, Rotation, and Scaling. The translation rate is expressed in meters per second. The rotation rate is expressed in degrees per second. Finally, the scaling rate is in units per second. When not specified, a default rate of zero is assumed.

For instance, a primary radar antenna that rotates at a rate of 10 degrees per second about its Z-axis would require the following XML tags:

```
<CDB:Articulation name="Primary Radar 1" id="5376">
  <Rotation>
    <Z rate="10"/>
  </Rotation>
</CDB:Articulation>
```

Another example, to illustrate how to attribute a rotating wind mill; assuming the mill rotates about the Y-axis at a rate of 5 degrees per second:

```
<CDB:Articulation>
  <Rotation>
    <Y rate="5"/>
  </Rotation>
</CDB:Articulation>
```

Gimbal limits are mandatory on DOF nodes and the appropriate flags must be set to specify which degrees of freedom are controlled by a particular articulation. The Flags field is located at offset 376 in the OpenFlight DOF record and its value cannot be zero because the articulation must control at least one degree of freedom.

## 6.10.2 Usage

### 6.10.2.1 Rotating Parts

A common problem in simulation is to correlate the linear speed of a model with the angular speed of its wheels. More generally, the client device simulation models often require the dimension of rotating parts. This information can be obtained from the zone extent; the bounding box surrounding a zone provides the dimension of rotating parts.

## **6.11 Model Light Points**

The CDB Specification does not make a distinction between light points and light sources. Both represent real lights that emit light and that can illuminate neighboring objects. In most current visual systems, a light point is a simple representation of a point source of light when viewed from a distance; it has no observable lighting effect on its immediate surroundings. In real-life however, as an observer moves closer to the light, its lighting or illuminating effect on the surrounding objects becomes increasingly observable; furthermore, the actual shape of the light also becomes more distinct.

In a typical simulator, client-devices may choose to limit the representation of the light to a single point and neglect the illuminating effect of the light on neighboring objects and terrain. For this reason, it is up to the client (and its RTP) to determine whether a light can illuminate its surroundings or not; the decision is based on the type of light and the inherent capabilities/capacity of the client.

Another point to consider is the fact that a light may have a very different representation depending on the client device. For instance, consider the visual representation of a light by an IG compared with the representation required by a radar system, NVG device or a FLIR device.

For all of these considerations, the CDB Specification has adopted the following approach in defining lights. The OpenFlight file defines only the position, direction and the name of the light type; no other attributes are specified. The CDB Specification provides a very elaborate light type naming convention. This convention permits clients to internally derive all of the properties and parameters needed to render the light. The approach is entirely device-independent. Modelers need not concern themselves with hundreds of parameters, many of which are often specific to underlying algorithms within the client. The naming convention ensures that the client has all of the information needed to capture the modeler's intent. Because the approach is device-independent, the rendering is limited only by the client's capabilities, not by the database itself.

As a result, lights in OpenFlight are defined by inserting an Indexed Light Point record into the OpenFlight scene graph. A vertex and a normal are stored in a Vertex List record that defines the position and direction of the light. The name of the light type is stored in the Light Point Appearance Palette record.

The light type's name fully defines the appearance, animation and other characteristic relevant to the field of simulation. It is the responsibility of the client to supply the internal parameters that correspond to each of the light types supported by the CDB Specification.

Light type naming conventions are defined in Section 2.3, Light Naming, and the list of names is presented in Appendix E.

A Vertex List record must follow the OpenFlight Indexed Light Point record. The list of vertices contains one vertex if a single light point is defined. The list contains

several vertices when multiple independent light points are defined. An optional matrix and replication count permits the definition of a light string.

**Table 6-33: OpenFlight Records for a Light Point**

INDEXED LIGHT POINT MATRIX (optional) REPLICATE (optional) PUSH LEVEL VERTEX LIST POP LEVEL
--

## 6.12 Model Attributes

This section defines a general attribution mechanism to add CDB and Vendor-specific attributes to any OpenFlight nodes. These attributes follows the rules of inheritance; they are automatically propagated from higher levels through lower levels of the OpenFlight graph. A child node inherits the attribution of its parent node.

### 6.12.1 Definition

Model attributes are added to OpenFlight nodes through a Comment record containing XML tags. The general format is as follow:

<pre>&lt;CDB:node name="..."&gt;   &lt;ns:Attribute name="..." value="..." /&gt;   ... other attributes &lt;/CDB:node&gt;</pre>
---

`<CDB:node>` identifies the node to which the attributes are added. The `node` token can take the following values:

- Zone
- Point
- Group
- Object
- Switch
- Face
- Mesh
- Articulation
- Light
- XRef
- LOD

The XML namespace (`ns`) of the attribute is optional; when present it identifies the owner of the attribute. When not specified, the default namespace is CDB.

Any CDB Attributes that are listed in section 5.7.1.3 can be used as node attributes. The name of the attribute is the key to search for the matching symbol into the



metadata file named CDB\_Attributes.xml; this file is described in section 5.1.7 and provides the means to interpret the value of the attribute.

### 6.12.2 Vendor Attributes

A vendor attribute is identified by its XML namespace. The Specification uses the CDB namespace; a vendor may use any other string to identify itself. The definition of vendor attributes must be stored in Vendor\_Attributes.xml.

It is understood that vendor attributes are not interpreted by any other client-devices other than those supported by the vendor. Reliance by a vendor on Vendor Attributes can reduce the interoperability of the CDB with other vendors.

### 6.12.3 Examples

To add the LPH attribute to a CDB Light node, use the following comment:

```
<CDB:Light>  
  <Attribute name="LPH" value="300"/>  
</CDB:Light>
```

Assume a T2DModel contains the Los Angeles International Airport as one of its 2DModels; the zone associated with the airport could use the APID attribute in the following manner:

```
<CDB:Zone name="Los Angeles International Airport">  
  <Attribute name="APID" value="KLAX"/>  
</CDB:Zone>
```

A company named “Acme Inc.” uses the string “Acme” as the namespace qualifying its vendor-specific attributes. If the company wants to add the MyAttr attribute to a CDB Articulation, it could do so by using the following XML tags:

```
<CDB:Articulation name="Primary Gun 1" id="4416">  
  <Acme:Attribute name="MyAttr" value="-1.23"/>  
</CDB:Articulation>
```

To interpret the attribute, a client application searches the file Vendor\_Attributes.xml for an attribute whose symbol is MyAttr. When found, the application knows how to parse and interpret the attribute’s value. Furthermore, if the client application recognizes the identification of the vendor (Acme:), it knows what to do with MyAttr.

## 6.13 Model Textures

To achieve a certain degree of realism, models require the use of textures. Furthermore, textures add details to a model without increasing its polygon count. This is excellent to reduce the complexity of the geometry but at the same time, it creates a load management issue for client devices that are interested in these textures. In the case of GTModels and MModels, textures are separate files that must be loaded after the model geometry files are read and loaded by client devices; in the case of GSModels and T2DModels, the textures can be loaded concurrently with the model geometry files. A client device discovers the existence of textures while loading the model.

One of the goals of the CDB Specification is to allow client devices to implement efficient load management mechanisms. For this reason, the Specification decouples as much as possible the texture aspect of a model from its geometry aspect. This is done by storing all textures related to Models in separate directories.

Recall that the texture filenames itself are constructed from the dataset number, the texture type (selectors 1 and 2), and the texture LOD and these are then concatenated to a modeler-specific texture name. Section 6.13, Model Textures, provides a description and usage of all of the CDB texture types for Models. The values of component selectors 1 and 2 convey a semantic meaning to the texture (time-of-year, paint scheme, night map, light map, normal map, etc) and determine whether the texture is to be used as base texture or as a subordinate texture and whether the texture is switchable (described in the next section).

### 6.13.1 Handling of Multi-textures

In OpenFlight, several types of textures can be applied in various combinations. Textures fall in two broad categories: Base and Subordinate.

#### 6.13.1.1 Base Texture Layer

Base textures<sup>63</sup> are set of mutually exclusive model textures that provide texture color/intensity modulation for the model. While a model can have many base textures, only one base texture can be referenced and applied to model geometry at a time.

The CDB Specification supports the following type of Base Textures:

- (1) ***Year-Round Texture:*** A year-round texture used with GTModels, MModels, GSModels, T2DModels . In the case of MModels, base-textures are often replaced with an appropriate Paint Scheme texture (Uniform, Camouflage or Airline).
- (2) ***Quarterly Textures:*** A set of 4 textures, each representative of a quarter within the calendar year used with GTModels, GSModels, T2DModels . The textures

---

<sup>63</sup> The CDB Specification uses the term “base texture” the same way as OpenFlight and Creator do.

must be provided as a complete set, i.e., it is assumed that all 4 textures of the same kind (i.e., all four textures have their component selector 1 set to 003) and are all present in the model's texture directory. The presence of a quarterly texture reference in model geometry tells the client-device that a quarterly texture set is available. This allows the client-device to select any one of the available 4 textures at rendering time. Only one of the textures need be referenced by the OpenFlight scenegraph geometry, preferably the third quarter texture. It is also assumed that all 4 textures share the same UV mapping.

- (3) **Monthly Textures:** A set of 12 textures, each representative of a month within the calendar year used with GTModels, GSModels, T2DModels. The textures must be provided as a complete set, i.e., it is assumed that all the 12 textures are of the same kind (i.e., all twelve textures have their component selector CS1 = 002) and are all present in the model's texture directory. The presence of a monthly texture reference in model geometry tells the client-device that a monthly texture set is available. This allows the client-device to select any one of the available 12 textures at rendering time. Only one of the textures need be referenced by the OpenFlight scenegraph geometry, preferably the June texture. It is also assumed that all 12 textures share the same UV mapping.
- (4) **Uniform Paint Scheme Textures:** Used on MModels with relatively uniform paint schemes should make use of this texture kind. Colors are listed in Appendix O. It is also assumed that all textures share the same UV mapping.
- (5) **Camouflage Paint Scheme Textures:** Used on MModels with camouflage paint schemes should make use of this texture kind. Camouflages are listed in Appendix O. It is also assumed that all textures share the same UV mapping.
- (6) **Airline Paint Scheme Textures:** Used on MModels that represent commercial aviation airliners should make use of this texture kind to implement the airlines paint scheme and logos. This base texture addresses the need for multiple skins painted on identical aircraft type. For instance, the B767-300ER is operated by more than 60 airlines throughout the world. Appendix O provides a complete list of Airliners. It is also assumed that all textures share the same UV mapping.
- (7) **Shadow Map Textures:** Used on MModels as pre-computed orthographic projections of the MModel. These textures are base textures used to accelerate the rendering of MModel shadows. Shadow map usage conventions are described in section 6.13.5.1, Model Shadow Textures.
- (8) **Motion Blur:** Used on MModels as pre-computed motion blurred textures of rotating parts (e.g., rotor disks). These textures are base textures used to aid client-devices in eliminating temporal aliasing artifacts. Motion blur textures conventions are described in section 6.9.2.3, Temporal Anti-aliasing.

### 6.13.1.2 Subordinate Texture Layer

Base textures can be supplemented with one or more<sup>64</sup> subordinate textures. Subordinate textures form a set of model textures that can be used to provide

---

<sup>64</sup> OpenFlight natively permits up to seven subordinate textures for a total of eight textures including the base texture.

additional color/intensity modulation or illumination modulation detail to the Base texture.

The CDB Specification supports the following types of subordinate textures:

- (1) **Night Map:** This subordinate texture is used to represent changes to models in their night configuration, typically as a result of lighting effects emanating from inside the model through windows. Night map textures conventions are described in greater detail in section 6.13.5.3, Model Night Maps.
- (2) **Detail Texture (Micro/Macro):** This subordinate texture is used to add details to a base texture that lacks the necessary resolution to provide the correct depth perception. Detail textures conventions are described in greater detail in section 6.13.5.6, Model Detail Texture Maps.
- (3) **Contaminants:** These textures are used to simulate thin layers of matter that accumulate on surface top. Contaminant textures conventions are described in greater detail in section 6.13.5.7, Model Contaminant and Skid Mark Textures.
- (4) **Normal Map:** Normal mapping is a technique used for faking the lighting of bumps and dents; when used in conjunction with a render's light sources, it can add surface detail without using more polygons. This subordinate texture is a 3-component texture that encodes the normals at each texel. Tangent-space normal maps conventions are described in greater detail in section 6.13.5.5, Model Tangent-space Normal Maps.
- (5) **Reflection Map:** Conventions are described in detail in section 6.13.5.8, Model Cubic Reflection Maps.
- (6) **Light Map:** This subordinate texture is used to represent the effect of external light sources onto a model. Light map textures conventions are described in greater detail in section 6.13.5.4, Model Light Maps.
- (7) **Gloss Map:** A texture that describes whether a surface is matte or gloss; described in section 6.13.5.9, Model Gloss Maps.
- (8) **Material Texture:** To specify the composite materials at the level of a single texel; described in section 6.13.5.10, Model Material Textures.

Client-devices are required to use the modeler supplied layer number to determine the order in which the subordinate textures are to be rendered. The base layer is always rendered first, followed by subordinate layer 1, 2, 3, etc. Gaps within the layer sequence are permitted.

Note that layer numbers are not assigned nor reserved to specific subordinate textures.

### 6.13.1.3 Texture Mapping Conventions

The following table provides the texture mapping for use with each kind of textures.





Base Texture		Subordinate Texture	
Kind	Mapping	Kind	Mapping
001	Modulate	051	Decal
002	Modulate	052	N/A
004	Modulate	053	Modulate
005	Modulate	054	Modulate
006	Modulate	055	Modulate
007	Modulate	056	Add
008	Modulate	057	N/A
009	Modulate	058	N/A

### 6.13.2 Default Gamma Corrections

The default gamma corrections of 3D model texture datasets are defined by the following set of parameters found in the Defaults.xml metadata file.

- Default\_GSModelTexture\_Gamma
- Default\_GSModelInteriorTexture\_Gamma
- Default\_GTModelTexture\_Gamma
- Default\_GTModelInteriorTexture\_Gamma
- Default\_MModelTexture\_Gamma

If a parameter is not found in Defaults.xml, or if Defaults.xml is not found in the metadata directory, assume a default gamma correction of 1.0.

See Appendix S for the complete list of default parameters.

### 6.13.3 Texture Dimension

It is generally accepted by the modeling community to limit texture dimensions to a power of 2. The CDB Specification goes a step further and enforces this practice.

To preserve the original texture resolution as much as possible, it is suggested to resize the source texture to the nearest<sup>65</sup> power of 2. For instance, if a source texture measures 72 pixels wide by 13 pixels high, it is recommended to resize it to 64 by 16 pixels.

$$\text{Texture Dimension} = 2^n \times 2^m$$

Where  $n$  and  $m$  are positive integers ( $n, m \geq 1$ ).

#### 6.13.3.1 Texture Mipmap

The CDB Specification demands that mipmaps associated with a given texture be present in the texture directory. Furthermore, the Specification requires that mipmaps be stored in individual files.

<sup>65</sup> Note here that we do not recommend resizing to the *next* power of 2; instead, resize to the *nearest* power of 2.

$$\text{Number of Mipmaps} = \max(n, m) + 1$$

For instance, a texture whose dimension is  $2^3 \times 2^4$  has a total of 5 mipmaps.

#### 6.13.3.2 Texture Size

The naming conventions of all model textures are described in Chapter 3. For texture file whose name uses the W field, the value of the field is a power of 2 representing the largest dimension of a (possibly rectangular) texture.

$$\text{Texture Size} = 2^W$$

Where  $W$  is a non-negative integer ( $W \geq 0$ ).

#### 6.13.3.3 Texel Size

For texture file whose name uses the L field, the value of the field is related to the size of the texels in accordance to Table 3-1: CDB LOD vs Model Resolution.

#### 6.13.4 Texture Palette

The OpenFlight Texture Palette record stores the names of all textures that are possibly referenced by the model; that includes all base and subordinate textures (i.e., all skins and all interchangeable textures). Each palette entry contains the path and filename of one texture. The CDB Specification demands that the path be relative to the OpenFlight file.

Below are examples of entries in the texture palette.

##### 6.13.4.1 MModel Example

In the case of a moving model, the OpenFlight file resides in the MModelGeometry directory; for instance, the M1A2 resides in

```
\CDB\MModel\600_MModelGeometry\1_Platform\1_Land  
  \225_United_States\1_Tank\1_1_225_1_1_3_0\
```

Its main texture is called M1A2 and resides in

```
\CDB\MModel\601_MModelTexture\M\1\M1A2\
```

The corresponding palette entry would be

```
..\..\..\..\..\601_MModelTexture\M\1\M1A2\  
  D601_S005_T001_W11_M1A2.rgb
```

##### 6.13.4.2 GTModel Example

In the case of a geotypical power pylon model, the OpenFlight file resides in the GTModel directory

```
\CDB\GTModel\510_GTModelGeometry\A_Culture\T_Comm  
  \040_Power_Pylon\Lxx\
```



Assuming its texture is called Pylon, it resides in

```
\CDB\GTModel\501_GTModelTexture\P\Y\Pylon\
```

The corresponding palette entry would be

```
..\..\..\..\..\501_GTModelTexture\P\Y\Pylon\  
D511_Sxxx_Txxx_Lxx_Pylon.rgb
```

#### 6.13.4.3 GSModel Example

In the case of a geospecific model, its OpenFlight file resides in the GSModelGeometry directory. An example is

```
\CDB\Tiles\lat\lon\300_GSModelGeometry\Lxx\Ux\
```

If the model refers to a geospecific texture, it resides in

```
\CDB\Tiles\lat\lon\301_GSModelTexture\Lxx\Ux\
```

The corresponding palette entry would be

```
..\..\..\..\301_GSModelTexture\Lxx\Ux\  
latlon_D301_Sxxx_Txxx_Lxx_Ux_Rx_TNAM.rgb
```

If the model refers to a geotypical texture, it resides in

```
\CDB\GTModel\501_GTModelTexture\T\N\TNAM
```

And the corresponding palette entry would be

```
..\..\..\..\..\GTModel\501_GTModelTexture\T\N\TNAM\  
D511_Sxxx_Txxx_Lxx_TNAM.rgb
```

#### 6.13.4.4 T2DModel Example

In the case of a tiled 2D model, its OpenFlight file resides in the T2DModelGeometry directory. An example is

```
\CDB\Tiles\lat\lon\310_T2DModelGeometry\Lxx\Ux\
```

If the model refers to a geospecific texture, it resides in

```
\CDB\Tiles\lat\lon\301_GSModelTexture\Lxx\Ux\
```

The corresponding palette entry would be

```
..\..\..\..\301_GSModelTexture\Lxx\Ux\  
latlon_D301_Sxxx_Txxx_Lxx_Ux_Rx_TNAM.rgb
```

If the model refers to a geotypical texture, it resides in

```
\CDB\GTModel\501_GTModelTexture\T\N\TNAM
```

And the corresponding palette entry would be

```
..\..\..\..\..\GTModel\501_GTModelTexture\T\N\TNAM\  
D511_Sxxx_Txxx_Lxx_TNAM.rgb
```

## 6.13.5 Usages

### 6.13.5.1 Model Shadow Textures

Ideally, Model shadows should be generated at runtime by the client-device from the model's actual geometry. However, depending on the technique used by the client device, special textures called projected shadow maps may be used to cast shadows from Models.

When the projected shadow map technique is used, special object nodes are used to store the shadow polygons.

#### 6.13.5.1.1 Shadow Geometry

When geometry exists for the purpose of casting shadows, it must be located under an object node whose Shadow flag is set.

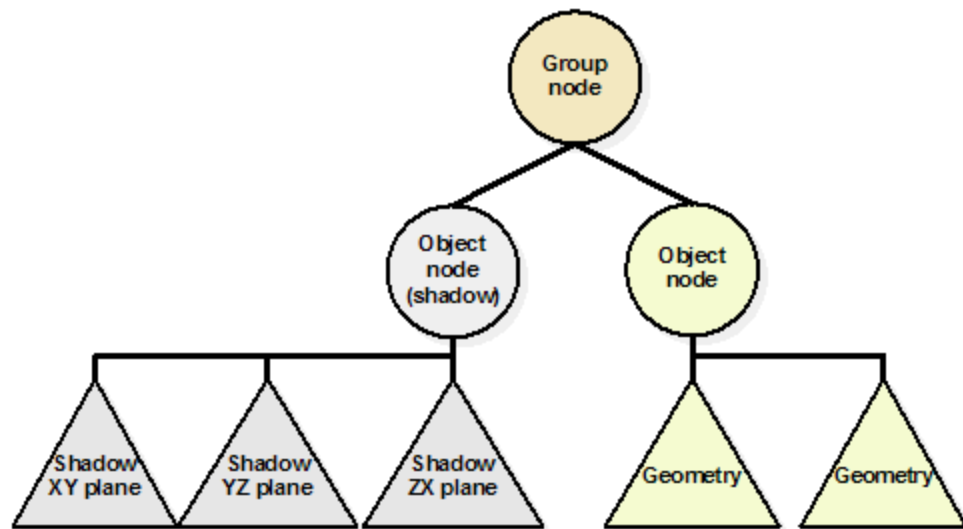


Figure 6-49: Using Shadow Polygons

Several object nodes can be used to store several polygons all textured with projected shadow maps. It may be desirable to also create separate shadow maps for major articulated parts and locate these shadow objects just under their corresponding DOF nodes.

#### 6.13.5.1.2 Shadow Maps

Projected shadow maps are created by applying one, two or three orthographic projections on the model (or optionally on major articulated parts of the model). Figure 6-50: Example of a Shadow Map in the XY Plane shows one example of a shadow map of an aircraft in the XY plane. Similar maps can also be produced for the YZ and ZX planes.



**Figure 6-50: Example of a Shadow Map in the XY Plane**

A projected shadow map is a monochrome (single-component) texture without transparency. It represents the mask to cut out the contour of the model. In theory, a black and white texture would be enough; however, shades of gray are permitted to represent semi-transparent surfaces that could be present on the model. In any case, a value of 0 (black) means the model does not block the passage of lights. The opposite value, 1 (white), indicates the model completely obstructs the light.

Because the shape of the model may change with damage states, each model state should have its own set of projected shadow maps. Section 6.9.2.2 describes damage states.

Shadow maps are general base textures. Their Texture Kind is 007 and their Texture Index is a sequence number when several shadow maps exist for the same Model.

To illustrate the naming convention, assume the shadow map from Figure 6-50: Example of a Shadow Map in the XY Plane, is called “aircraft”. According to Section 3.5.2.1, MModelTexture Naming Convention, and Section 5.5, MModel Library Datasets, the resulting file name would be:

```
D601_S007_T001_Wnn_aircraft.rgb
```

The value Wnn represents the texture size,  $2^{nn}$ , and is explained in Section 6.13.3.2, Texture Size.

Note that if a client-device generates shadows on its own, without the support of pre-computed projected shadow maps, it can ignore all OpenFlight object nodes whose Shadow flags are set as well as all textures associated with these nodes.

#### 6.13.5.2 Model Skin Textures

Models skins are base textures that correspond to one or more moving models paint schemes or one or more time-of-year representations of the cultural feature.

For instance, the same tank can be painted with several different colors to match various areas of operation. Below are two examples of the same tank, the M1A2 Abrams, painted for operation in a desert area (Figure 6-51) or in a forest area (Figure 6-52).



**Figure 6-51: The M1A2 Abrams with a Desert Camouflage**



**Figure 6-52: The M1A2 Abrams with a Forest Camouflage**

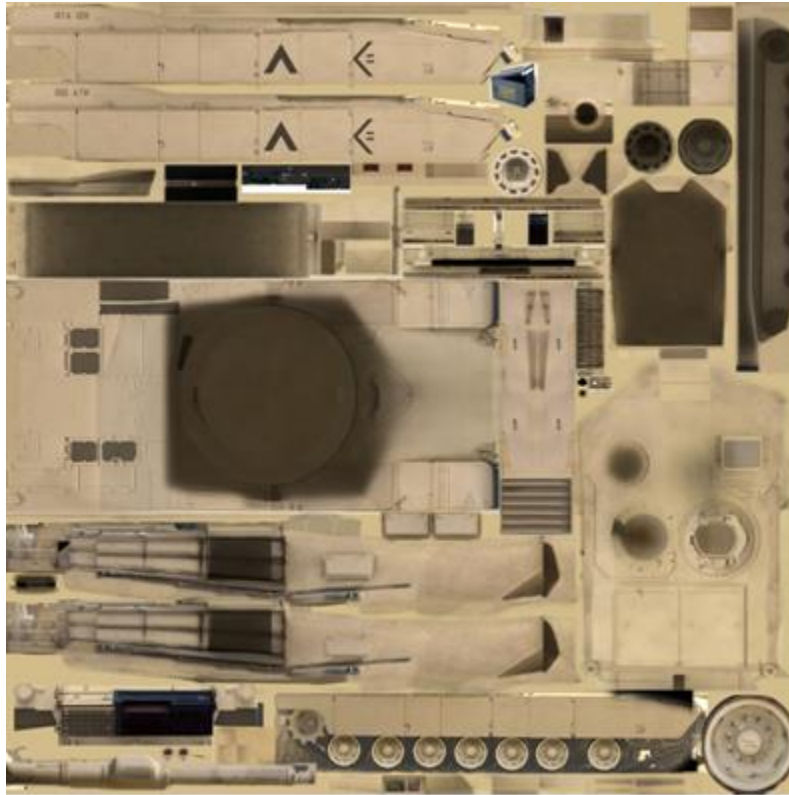
The two different textures for this tank qualify for use as skins since they have been designed in such a way that they can be exchanged for one another without affecting their mapping on the affected polygons. The mapping of both textures must be identical since only the texture is changed, not the UV mapping.

OpenFlight itself does not provide an explicit mechanism to change the base texture assigned to faces. In fact, OpenFlight supports a single base texture per face record. The other textures that can be added to a face are called layers and none of them is a replacement for the base texture.

In order to have several skins for a single model, the CDB Specification provides the mechanism defined in 6.14.5.2, Texture Switch. The enumeration values for each skin can be found in Appendix O.

The M1A2 used in the above examples has two skins. Each skin is made of a single texture that happens to be a mosaic of all the individual textures used by the model. Figure 6-53: M1A2 Desert Skin Mosaic below shows one of the M1A2 skins.





**Figure 6-53: M1A2 Desert Skin Mosaic**

The following texture kinds implement the concept of model skins:

- Kind 002 – Monthly Representation
- Kind 009 – Quarterly Representation
- Kind 004 – Uniform Paint Scheme
- Kind 005 – Camouflage Paint Scheme
- Kind 006 – Airline Paint Scheme

Paint schemes apply to moving models only. Appendix O lists available paint schemes.

Time-Of-Year representations are appropriate for cultural features. A good example is the case of leafy trees. Depending on the hemisphere and the latitude, several textures represent leafy trees at different stages during the year.

All texture kinds listed above are mutually exclusive; also, all instances of textures of a kind are mutually exclusive. Since all texture kinds above are base textures, and because only one base texture can be active at any one time on a face of a model, it follows that only one skin can be active at a time.



### 6.13.5.3 Model Night Maps

Night maps fall under the category of subordinate textures. Night and light maps (see next section) are both used at night to represent how interior and exterior light sources change the appearance of a Model. It is possible for a Model to have both a night and a light map, or just a light map. However, it is not possible to have a night map alone.

Simulator client-devices invoke a night map when the (simulated) light sources located inside the model need to change its appearance at night. This is the case when the interior light sources shine through openings like windows and portholes. To simulate the effect of lights emitted through these openings, a night map is created; it adds these bright window details normally missing from the base day texture.

The creation of a night map for models is left to the discretion of the modeler. Creating additional geometry for the windows and changing the material associated with the polygons to incorporate an emissive component can also produce a lighting effect similar to night maps. However, this approach requires additional (unnecessary) model geometry that adds additional computational load in the client-devices. For this reason, the use of night maps is recommended.

Light maps differ from night maps in that they combine the effect of exterior lighting with interior lights. A light map acts as a (colored) filter to mask portions of the model that are no longer visible at night when no ambient light exists.

A Model may have a relatively different aspect at night. This difference comes from two changes in the environment. The ambient illumination provided by sunlight is totally absent at night; only the moon and man-made light sources affect the appearance of objects. When present, the moon provides only a modest level of illumination when compared to the sun. In addition, the model itself might have internal lights turned on that are not modeled in day version of the texture but that do affect its appearance at night.

To illustrate these differences, imagine a building as seen during the day. No light seems to come out of its windows because the average daytime sunlight overwhelms any man-made lighting (internal to the building and coming out of the windows). At night, the outside walls of the building have not changed but light is now emanating from the windows. This is an important change that requires a modification to the texture used to represent the walls.

Another example of the use of a night map is the case of an aircraft flying at night. During daytime, the aircraft windows look dark while at night, light comes from the inside and the windows appear white.

Night maps are used to add details to base textures; details that are not visible during the day and that become visible at night. Therefore, a night map is not a replacement for the base texture. It is used in conjunction with the base texture.

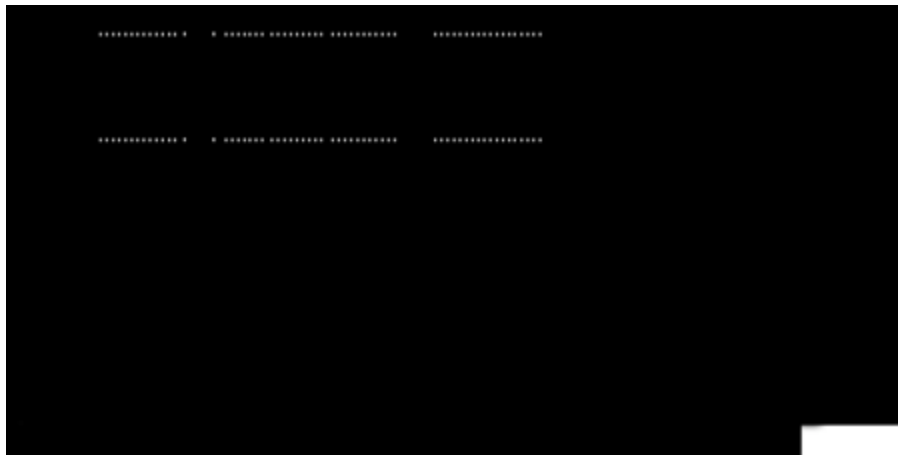
The next figures illustrate the purpose of night maps.

Figure 6-54: Base Texture is the base texture used in modeling a commercial aircraft, the Airbus 330, during the day. Notice that portholes are represented by dark rounded rectangles because the lights in the cabin are off. The same is true for the cockpit windows located in the bottom right of the texture.



**Figure 6-54: Base Texture**

Figure 6-55: Night Map, is the model's corresponding night map and shows the same portholes but this time brighter to reflect the fact that cabin lights are on. This time, notice the appearance of the cockpit windows as well as the presence of colors in them. Remember that this texture is used to add details that may be missing from the base texture.



**Figure 6-55: Night Map**

There are constraints imposed on night maps:

- A night map must have the same size as its base texture.
- A night map uses the same UV mapping as its base texture.
- A night map has a similar format as its base texture (RGB or Intensity) plus an alpha channel.

#### 6.13.5.3.1 Night Map Generation

A night map is mapped on top of its base texture using a Decal texture environment. Since a night map is a subordinate texture, it is mapped on polygons using an OpenFlight multitexture record. The Effect field of this multitexture record must contain the value 0 indicating to use the Texture Environment mapping defined in the Texture Attribute file. The Environment Type field found in the Texture Attribute file must contain the value 2 indicating a Decal environment mapping.

The night map alpha channel is in fact a mask identifying which portions of the base texture are replaced by the night map. Accordingly, the alpha channel contains a value of 0 when the corresponding texel of the base texture is left intact. However, the alpha channel will contain the value 1 when the corresponding texel of the base texture is replaced by the equivalent night map texel. Overall, the Decal environment mapping applies the following transformation to the base texture.

$$C = C_b \cdot (1 - A_n) + C_n \cdot A_n$$

where...

$C_b$  is the color component (or intensity) of the base texture

$C_n$  is the color component (or intensity) of the night map

$A_n$  is the alpha component of the night map

Since the values found in the night map alpha channel are limited to 0 and 1, the resulting color will either be the one found in the base texture when  $A_n$  is 0 or the one found in the night map when  $A_n$  is 1.

#### 6.13.5.4 Model Light Maps

Light maps also fall under the category of subordinate textures. Night maps and light maps are both used at night to represent how interior and exterior light sources change the appearance of a Model. It is possible for a Model to have both a night map and a light map, or just a light map. However, it is not possible to have a night map alone.

Light maps differ from night maps in that they combine the effect of exterior lighting with interior lights. A light map acts as a (colored) filter to mask portions of the model that are no longer visible at night when no ambient light exists.

A light map is used when active light sources are located on the outside of the model. This technique is used to simulate the appearance of a model when lit by local spotlights. For instance, spotlights may be used to illuminate a building at night. The light map provides the illumination pattern that represents the spotlight illumination on the building. The technique provides a convenient mean to produce interesting and entirely predictable lighting effects without resorting to computationally intensive local light sources. Their effects are already incorporated into special textures called light maps.

A light map also contains a mask related to the night map when present. Remember that a light map is a filter (a mask) to retain the detail associated with the base texture and its optional night map.

As opposed to a night map, a light map does not have constraints. More specifically:

- A light map does not need to be of the same size as its base texture.
- A light map has its own UV mapping.
- A light map can be an intensity map or an RGB image.

Note that when light sources are modeled with light maps, they only affect the model onto which they are applied.

The next set of figures illustrates how light maps contribute to the lighting of a model. Note that a light map is not applied directly to the model base texture. The light map is first modified to take into account the ambient lighting, and then the resulting lighting is applied to the model.

Figure 6-56: Light Map, is the light map matching the base texture in Figure 6-54: Base Texture. Notice that it combines light lobes representing external light spots with the mask associated with internal light sources from the night map. This mask is used to key in details that stay visible at night.



**Figure 6-56: Light Map**

Figure 6-57: Combined Effect of Base Textures and Light Maps, shows on the actual aircraft the result of applying the light map from Figure 6-56: Light Map to the base texture from Figure 6-54: Base Texture. Notice that portholes and cockpit windows are still dark since the base texture has not been modified by the night map yet.



**Figure 6-57: Combined Effect of Base Textures and Light Maps**

Figure 6-58: Combined Effect of Night and Light Maps, shows the result of adding the night map to the base texture and then applying the light map. This time, we can clearly see the lights coming through portholes and cockpit windows.



**Figure 6-58: Combined Effect of Night and Light Maps**

#### **6.13.5.4.1 How and When to Use Night Maps and Light Maps**

The CDB Specification recommends the use of night maps to represent lights that are internal to the model; this permits the client device to control the appearance of the model with internal lights on or off. This condition is usually true at night, hence the name of the texture.

Similarly, the CDB Specification recommends the use of light maps to represent the effect of lights that are external to the model; this permits the client-device to control the appearance of the model with external spotlights on or off.

Note that night and light maps can be applied to any of the skins since skins are base textures.

#### **6.13.5.4.2 How and When Not to Use Light Maps**

A client device may discard light maps if the effect of external lights is internally generated by its GPU. It can be envisioned that future development of specialized hardware – such as graphics processor unit – will allow more of the lighting effects to be generated in real-time. When this time comes, artificial textures generated off-line such as light maps will become obsolete.



#### 6.13.5.5 Model Tangent-space Normal Maps

A normal map is an RGB texture (without an alpha channel) where the normal to the surface is encoded in the Red, Green, and Blue channels. The normal (i, j, k) values are encoded in the following manner into the 8-bit value of each channel:

- $R [0, 255] = i [-1.0, +1.0]$
- $G [0, 255] = j [-1.0, +1.0]$
- $B [0, 255] = k [-1.0, +1.0]$

The mapping is identical on all channels; the range of all possible 8-bit values (0, 255) is mapped linearly to the range of floating point values -1.0 to +1.0. This mapping provides a resolution of  $2/255$  or 0.0078.

In addition, the reader should note that the floating-point value 0.0 has no exact integer equivalent<sup>66</sup>. Here, the closest value to 0.0 is approximately  $\pm 0.0039$  and is obtained when the channel contains 127 or 128.

Besides this particular encoding of the normal into the RGB channels, a normal map has all the other attributes of a standard RGB texture whose format is defined in Appendix P, SGI Image File Format.

In the industry, there are at least two types of Normal Map: object-space normal map, and tangent-space normal map. Both types have their pros and cons. The CDB Specification opts for tangent-space normal map. A sample is shown here.

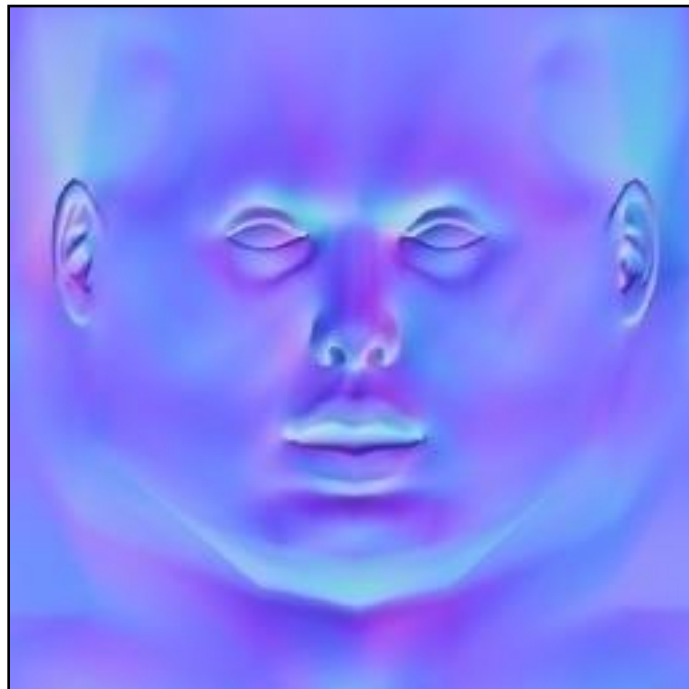


Figure 6-59: Normal Map Sample

<sup>66</sup> The conventional OpenGL mapping specifies that -1 and 1 can be represented exactly, but 0 can not.



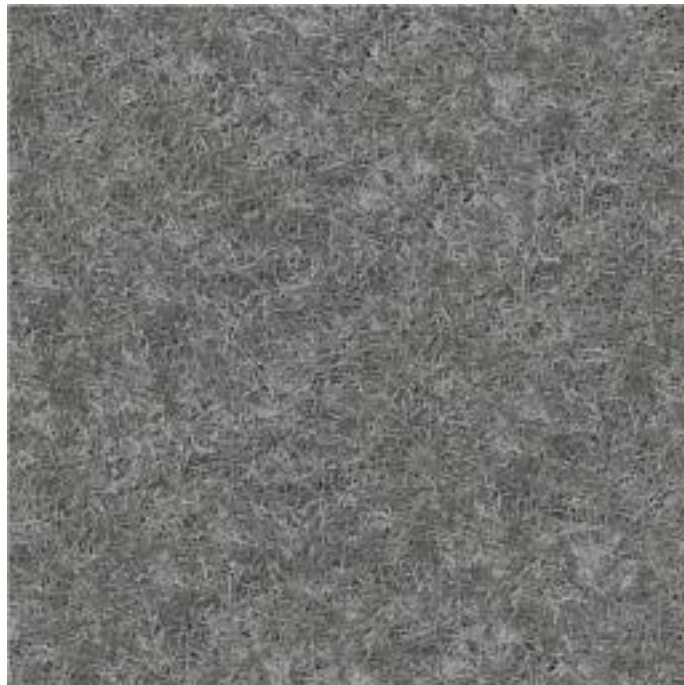
Typically, the normal points away from the surface, and not toward the underlying surface. For this reason, the value of the k-component of the normal is positive, most of the time, resulting in a bluish tint of the map. A negative k-component could indicate the presence of a cliff with an overhang, for instance.

#### **6.13.5.6 Model Detail Texture Maps**

A detail texture map is 1- or 3-component (aka channel) texture where each texel is represented as an 8-bit unsigned integer. A detail texture exhibits two important properties; it has a neutral luminance (intensity) and chrominance (color). This is achieved by applying the following constraints:

- The 8-bit unsigned value of each texel is scaled to a floating point value in the range -1.0 to 1.0
- The average value of an individual component is always 0.0
- The Detail texture is mapped on the underlying surface through a simple addition operation

The net effect of applying a Detail Texture Map is to highlight ( $> 0$ ) or darken ( $< 0$ ) fragment details on the underlying surface. When using a single component detail texture map, only the intensity of the resulting image is affected; when using a 3-component detail texture map, the color is also varied.



**Figure 6-60: Detail Texture Map Sample**

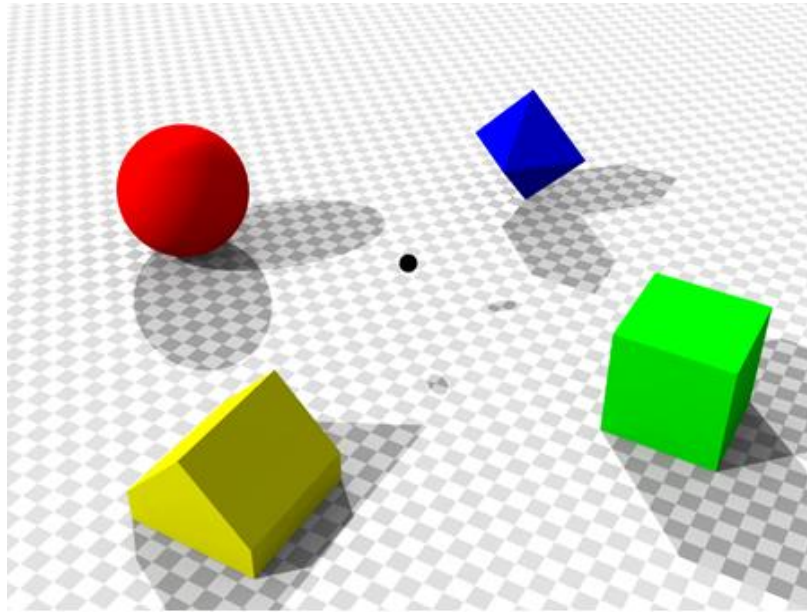
Recall that a detail texture map is a mean of adding high-frequency (spatial) details to a rather low-frequency image.

#### **6.13.5.7 Model Contaminant and Skid Mark Textures**

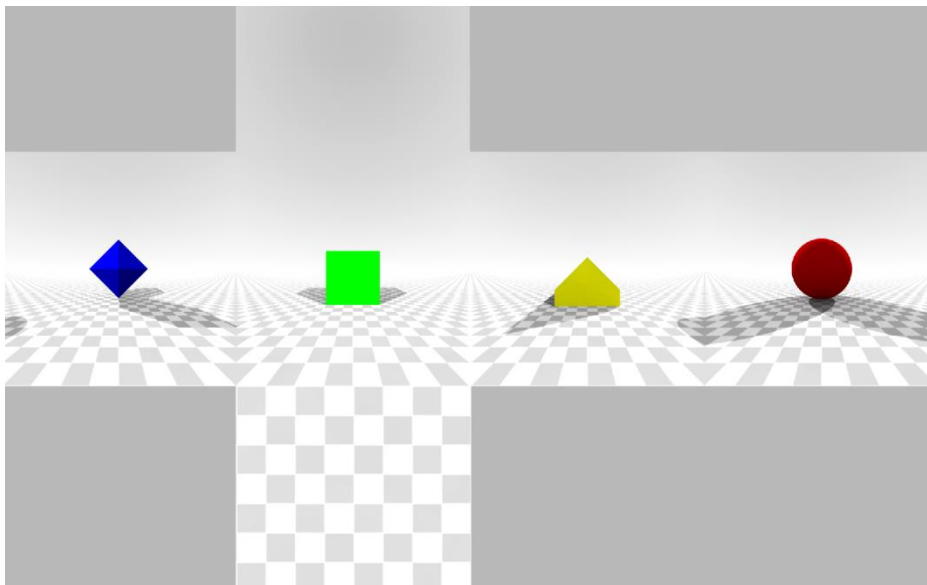
Historically, Image Generators of civil aviation simulators provided the means for flight instructors to control the appearance of airport runways, taxiways, and roads with various surface contaminants. To this end, the CDB provides a set of standardized Model Contaminant and Skid Mark Textures that are commonly used in flight simulators and listed in Appendix O. These textures are typically four-component (R, G, B, alpha) textures that act as an overlay to airport surfaces.

#### **6.13.5.8 Model Cubic Reflection Maps**

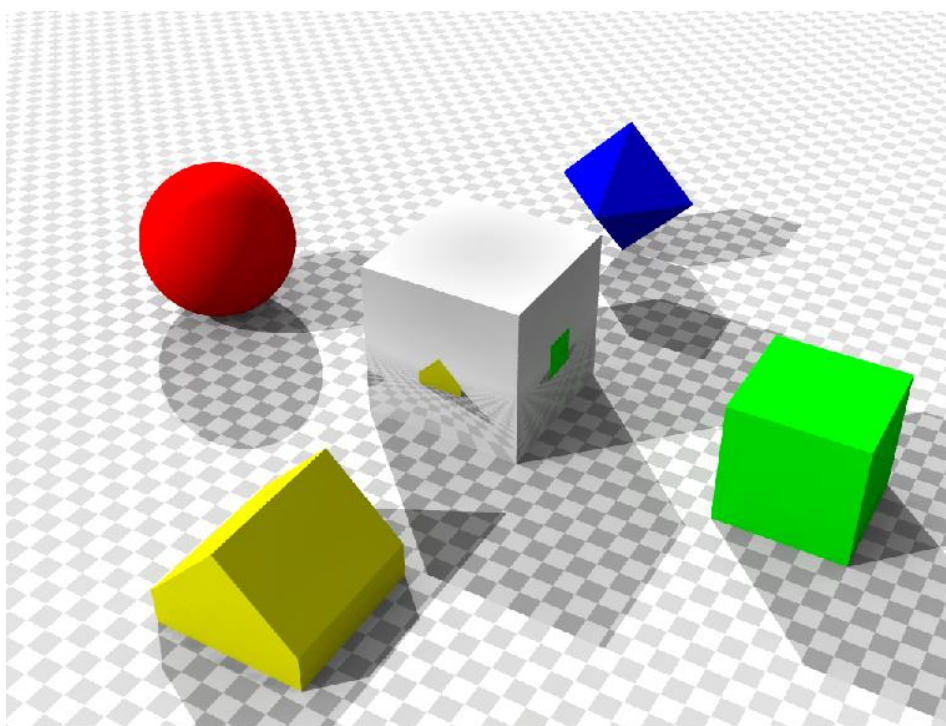
Reflection mapping (aka environment mapping) is an efficient image-based lighting technique for approximating the appearance of a reflective surface by means of a precomputed texture image. The texture is used to store the image of the distant environment surrounding the rendered object.



**Figure 6-61: Environment Used to Produce Reflection Map**



**Figure 6-62: Resulting Reflection Map**



**Figure 6-63: Rendered Reflection Map onto Reflecting Cube**

The CDB Specification assumes that the surrounding environment is stored using a cubic mapping approach. In this technique, the environment is projected onto the six faces of a cube and stored as six square textures or unfolded into six square regions of

a single texture. The reflection mapping approach is more efficient than the classical ray tracing approach of computing the exact reflection by tracing a ray and following its optical path. The reflection color used in the shading computation at a pixel is determined by calculating the reflection vector at the point on the object and mapping it to the texel in the environment map. This technique often produces results that are superficially similar to those generated by raytracing, but is less computationally expensive since the radiance value of the reflection comes from calculating the angles of incidence and reflection, followed by a texture lookup, rather than followed by tracing a ray against the scene geometry and computing the radiance of the ray, simplifying the GPU workload.

Note however that in most circumstances, a mapped reflection is only an approximation of the real reflection. Environment mapping relies on four assumptions:

- All radiance incident upon the *statically-positioned* object being shaded comes from an *infinite distance*. When this is not the case, then a) the reflection of nearby geometry appears in the wrong place on the reflected object, and b) no parallax is seen in the reflection.
- The object being shaded is *convex*, such that it contains no self-interreflections. When this is not the case the object does not appear in the reflection; only the environment does.
- The environment map is *valid for the location* for which it was generated.
- The environment is *static*.

#### 6.13.5.9 Model Gloss Maps

A gloss map is a texture that describes whether a surface is matte or gloss. The texture is used to modulate specular highlights in the same way the material shininess does. A gloss map is stored as an 8-bit single channel texture (a grey-scale image) where texels are mapped to the range 0.0 (matte) to 1.0 (glossy). The values in the gloss map play the same role as the single shininess value found in the OpenFlight material assigned to a polygon. In this way, the gloss map can effectively modulate the specularity on a per-pixel basis. Note that if the material applied to the surface has no specular component, then the gloss map has no effect.

#### 6.13.5.10 Model Material Textures

Material textures fall under the category of subordinate textures. They are mapped to Models the same way as any other textures. As such, the surfaces these textures are mapped to possess their own set of UV mapping.

A material texture tells the interested client devices (e.g., FLIR, CGF) what the underlying surface is made of. For this reason, a material texture is not at all related to a base texture. The two are completely independent and exist separately. A material texture does not require that a base texture be applied to the model. In fact,

it is perfectly possible to create a Model that does not use texture except for a single material texture describing its various materials.

The <Material> tag presented in section 6.5.3 is a high level mean of providing material information about the geometry of a model. With the use of a material texture, the modeler can provide highly detailed material information about the same model.

In short, the <Material> tag supports a polygon-based approach of sensor client devices such as FLIR, NVG, and RADAR. A Material texture is a texel-based approach supporting an implementation of such client devices with a much higher resolution.

In the case of the Raster Material dataset (dataset code 005) applied onto the terrain, it is conceivable that multiple layers and mixtures of materials are required to represent the rich variety of materials found on the earth surface. However, for Models, a single material layer is probably adequate for the vast majority of man-made objects.

## **6.14 Model Descriptor (Metadata) Datasets**

Each type of 3D Models has its set of ModelDescriptor datasets; they are:

1. GSModelDescriptor
2. GSModelInteriorDescriptor
3. GTModelDescriptor
4. GTModelInteriorDescriptor
5. MModelDescriptor

Each file is needed to summarize and regroup the information concerning one portion of a model, its shell or its interior. The information are collected and stored in an XML file to help client devices implement efficient load management mechanism.

The format of the model descriptor file is as follows:

```
<Model_Metadata>
  <Name>...</Name>
  <Identification>...</Identification>
  <Mass>...</Mass>
  <Parts>...</Parts>
  <Textures>...</Textures>
  <Configurations>...</Configurations>
  <Composite_Material_Table>...</Composite_Material_Table>
</Model_Metadata>
```

### **6.14.1 Model Name**

The <Name> is an arbitrary string from the character set presented in section 2.2. This name is the human readable version of the model identification code that follows.

## 6.14.2 Model Identification

Models are either modeled representation of cultural features or moving models. In both cases, the CDB Specification has a unique way to identify them. For moving models, the identification scheme corresponds to their DIS entity type. For cultural features, their FACC code is used.

### 6.14.2.1 Moving Model Identification

The DIS entity type is a list of up to seven integers and can be specified in two different manners. All fields have a default value of zero.

First, you can use a list of one to seven integers as illustrated here:

```
<Identification>
  <DIS_Entity_Type>
    <List>...</List>
  </DIS_Entity_Type>
</Identification>
```

Or you can use this more verbose syntax to specify the value of individual fields:

```
<Identification>
  <DIS_Entity_Type>
    <Kind>...</Kind>
    <Domain>...</Domain>
    <Country>...</Country>
    <Category>...</Category>
    <Subcategory>...</Subcategory>
    <Specific>...</Specific>
    <Extra>...</Extra>
  </DIS_Entity_Type>
</Identification>
```

All fields are limited to the range [0, 255] except the country code that can go up to 65535.

### 6.14.2.2 Cultural Feature Identification

For cultural features, their FACC code is specified in the following manner:

```
<Identification>
  <Feature_Attribute_Catalog_Code>
    <Code>...</Code>
    <Subcode>...</Subcode>
  </Feature_Attribute_Catalog_Code>
</Identification>
```

The FACC code has a fixed format of two letters followed by three digits; it is the same as the FACC attribute described in section 5.7.1.3.24. The subcode is an optional integer in the range [0, 999].

## 6.14.3 Model Mass

The model mass is optional. It makes sense only when the Model represents a moving model.





```
<Mass>
  <Total>...</Total>
  <Metal>...</Metal>
</Mass>
```

The total mass of the model is expressed in kilograms. The portion of the model that is made of a metallic alloy is expressed as a percentage of the total mass. The value of <Metal> lies in the range [0.0, 1.0].

When the model mass is specified, the total mass is mandatory while the metallic portion is optional. The total mass must be larger than zero. The metallic portion defaults to zero.

#### 6.14.4 Model Parts

A Model may be separated into several parts. If the complexity of a part justifies it, each part may be split into multiple files.

The whole section is optional. It is required only if more than one part exists or if a part has more than one file.

If present, the section is a list of at least one part formatted like this.

```
<Parts>
  <Part no="no" numFiles="numFiles" name="partName" />
  ...
</Parts>
```

The part number is mandatory. It starts at 1 and increases by 1 for each subsequent part. The first part is also referred to as the body of the model.

The number of files is optional and defaults to 1.

The part name<sup>67</sup> is optional and is used only to improve the readability of the file.

#### 6.14.5 Model Textures

This section lists all textures that could be possibly used by the model. In the event the model does not use texture, the whole section is omitted. The section contains a list of textures and optional texture switches.

---

<sup>67</sup> As a guideline, it is suggested to set the part name the same as the global zone name of that part. For instance, if the part represents an external fuel tank, a good name for both the part and its global zone would be “*External Fuel Tank*”.



```
<Textures>
  <Texture .../>
  <Texture .../>
  ...
  <Switch .../>
  <Switch .../>
  ...
</Textures>
```

#### 6.14.5.1 Texture Metadata

For each texture, the section provides the client device with the necessary information to decide when and which texture mipmap should be loaded.

The section is formatted like this.

```
<Texture no="number" name="name">
  <Dataset>...</Dataset>
  <Kind>...</Kind>
  <Index>...</Index>
  <Mipmap>min max</Mipmap>
  <Resolution>...</Resolution>
  <Coverage>
    <U>min max</U>
    <V>min max</V>
  </Coverage>
</Texture>
```

The texture number is a strictly positive integer to uniquely identify the texture. The texture name corresponds to the TNAM field in the texture filename as defined in Section 3.5.2.1, MModelTexture Naming Convention.

The <Dataset>, <Kind>, and <Index> fields correspond respectively to the dataset number and component selectors 1 and 2; they match the D, S and T fields in the texture filename.

The mipmap field defines the smallest and largest mipmap available for this texture. The value of this field is used to compose the W field in the texture filename of moving models (see examples in section 3.5.2.4).

The texture resolution is expressed in texels per meter<sup>68</sup>. It is the same for both the U and V axes even though it is recognized that it can differ between the two dimensions. The intent is to provide an indication of how precise the texture is when mapped to the model geometry. It helps client device decide which mipmap is more appropriate to use.

The texture coverage is optional and defines the minimum and maximum values for the U and V texture coordinates. This information indicates if the texture is repeated along one or both axes. If the coverage is in the interval [0, 1], the texture is clamped; otherwise, it is repeated.

---

<sup>68</sup> This unit of measurement (texels per meter) is akin to DPI (dot per inch) used to quantify the resolution of printers and displays.

### 6.14.5.2 Texture Switch

A Texture Switch is defined when switchable textures appear in the list of textures. Switchable textures are textures that can be exchanged for one another because they share the same UV mapping, as explained in section 6.13.5.2, Model Skin Textures.

The section is formatted like this.

```
<Switch no="number" name="name">
  <State no="number" name="name" textures="list"/>
  ...
</Switch>
```

The switch number is a unique positive integer identifying the switch. The switch name is a unique string limited to 32 characters; all switches are uniquely identified by a number and a name.

A switch has two or more states; each state selecting a list of one or more textures. State numbers are consecutive and start at 1. The state name is a unique string also limited to 32 characters. The list of textures associated with a state contains the texture numbers of the selected textures. Note that a state (e.g., a skin) may require more than one texture, hence the need to specify a list of textures associated with a state.

#### 6.14.5.2.1 Example

Assume that the following two textures are stored in the M1A2 texture folder:

```
\CDB\MModel\601_MModelTexture\M\1\M1A2\
D601_S004_T005_Wxx_M1A2.rgb
D601_S005_T001_Wxx_M1A2.rgb
```

Here is an excerpt of the model metadata presenting the two textures, the switch, and the two corresponding states.

```
<Textures>
  <Texture no="3" name="M1A2">
    <Dataset>601</Dataset>
    <Kind>4</Kind>
    <Index>5</Index>
    ...
  </Texture>
  <Texture no="10" name="M1A2">
    <Dataset>601</Dataset>
    <Kind>5</Kind>
    <Index>1</Index>
    ...
  </Texture>
  ...
  <Switch no="1" name="Paint Scheme">
    <State no="1" name="Uniform Beige Paint" textures="3"/>
    <State no="2" name="Desert Camouflage" textures="10"/>
  </Switch>
</Textures>
```

The texture switch is named “Paint Scheme” because it controls the selection of the paint scheme to apply to the M1A2. The first state selects texture 3 which

corresponds to a beige uniform paint; the second state selects texture 10 corresponding to a desert camouflage.

Note that the texture switch mechanism is not limited to base textures; it can be used to switch light maps for example.

### 6.14.6 Model Configurations

Often, a single Model – especially a moving model – comes with a variety of possible equipment and/or ordnance. This can be as diversified as fuel tanks, missiles, radio emitters, etc. To configure a model with its ordnance, the CBD Specification defines the concept of model configuration. A configuration defines the set of equipment and ordnance attached to the various stations found on the model.

The configuration section is optional. It is a list of one or more configurations defined like this.

```
<Configurations>
  <Configuration>...</Configuration>
  ...
</Configurations>
```

#### 6.14.6.1 Defining Stations in a Configuration

A configuration is a sequence of one or more stations, each defining one piece of equipment in one location.

```
<Configuration name="ConfigName">
  <Station name="StationName">
    <Location>...</Location>
    <Equipment>...</Equipment>
  </Station>
  ... other stations as needed
</Configuration>
```

The configuration and station names are both optional and are used for documentation purposes only.

The location of a station is defined by its fully qualified name as specified in section 6.5.5, Model Zone Naming.

#### 6.14.6.2 Defining Equipment in a Station

The equipment is defined by either its DIS identification or a reference to an external part, and an optional anchor point.

```
<Equipment name="EquipmentName">
  <Identification>...</Identification>
  <External_Part>...</External_Part>
  <Anchor>...</Anchor>
</Equipment>
```

The equipment name is optional and is used for documentation purposes only.

The anchor point is specified in the same manner as the location of a station, by providing its path (on the subordinate model) as specified in section 6.5.5, Model Zone Naming.

### 6.14.6.3 Defining Equipment Names

Either a DIS emitter name or a DIS entity type identifies the equipment. When the equipment is an emitter, the syntax is as follow.

```
<Identification>
  <DIS_Emitter_Name>...</DIS_Emitter_Name>
</Identification>
```

Emitter names are defined by the DIS standard. For DIS, refer to Section 8.1.1 of reference [4] for a list of DIS Emitter Names. For the HLA standard, the RPR-FOM lists all emitter names. To avoid confusion, both DIS and HLA refer to emitter names using numbers. For instance, the NATO emitter AS 15 KENT altimeter is referred to as emitter 8735.

When the equipment is another entity (e.g., a missile), its DIS entity type is supplied in the following manner.

```
<Identification>
  <DIS_Entity_Type>...</DIS_Entity_Type>
</Identification>
```

Recall that the DIS entity type is a list of up to 7 numbers as defined by reference [4]. For example, the AGM-114K-SAL Hellfire missile would be referred to as:

```
<DIS_Entity_Type>
  <List>2 2 225 1 3 5 1</List>
</DIS_Entity_Type>
```

or

```
<DIS_Entity_Type>
  <Kind>2</Kind>
  <Domain>2</Domain>
  <Country>225</Country>
  <Category>1</Category>
  <Subcategory>3</Subcategory>
  <Specific>5</Specific>
  <Extra>1</Extra>
</DIS_Entity_Type>
```

Equipment can also be defined by a reference to an external part if need be. A good example of such equipment is a fuel tank.

```
<External_Part>
  <Part_Number>...</Part_Number>
  <Configuration>...<Configuration>
</External_Part>
```

The external part is identified by its part number as defined previously in the <Parts> section.

The external part may also require it own configuration. Take the example of a Hellfire missile rack attached to an attack helicopter like the Apache. The rack can

hold up to 4 missiles. Each missile attaches to one of four separate weapon stations located on the rack. For this more complex example, assume the rack has only two missiles out of four. This configuration can be specified with the following piece of XML.

```
<External_Part>
  <Part_Number>1</Part_Number>
  <Configuration>
    <Station name="Missile 1">
      <Location>\Missile_Rack\Attach_Point[1]</Location>
      <Equipment>
        <Identification>
          <DIS_Entity_Type>
            <List>2 2 225 3 5 1</List>
          </DIS_Entity_Type>
        </Identification>
      </Equipment>
    </Station>
    <Station name="Missile 2">
      <Location>\Missile_Rack\Attach_Point[2]</Location>
      <Equipment>
        <Identification>
          <DIS_Entity_Type>
            <List>2 2 225 3 5 1</List>
          </DIS_Entity_Type>
        </Identification>
      </Equipment>
    </Station>
  </Configuration>
</External_Part>
```

With the help of model configurations, it is possible to create several variants of a single Model, each variant defined by its own configuration.

This way, one Apache can have two configurations, one when equipped with Hellfire missiles and one when equipped with rocket launchers.

#### 6.14.7 Model Composite Materials

The composite material table is the last component of the Model Metadata and is defined in section 2.5.2.2, Composite Material Tables (CMT).



## Chapter 7

### 7 CDB Radar Cross Section (RCS) Models

#### 7.1 Introduction

For devices such as Radars, a geometric representation of a model may often provide a level of fidelity which is insufficient or inappropriate for use in simulation or alternately, it may not be feasible to compute a radar cross-section of the model in real-time. Alternately, a user may wish to incorporate real-world RCS data into the simulator client-devices in order to further improve simulation fidelity. To this end, the CDB Specification defines a RCS (Radar Cross-Section) model representation for use by Sensor Simulation client-devices such as Radar and/or Sonar. It provides a signature model representing the overall relative reflectivity levels of a given Model Representation when viewed at discrete azimuth and elevation angles. The RCS data is then used in range and aspect calculations for the detection and classification of simulated targets (either ground or moving).

This chapter provides all of the information required to store RCS data within a CDB. Section A.7 of Appendix A provides a primer on radar, basic principles of operation and radar cross sections (RCS).

#### 7.2 RCS Data Model

This section concerns itself with the internal RCS model representation and its sub-structures, which forms a complete dataset layer.

##### 7.2.1 RCS Model Structure

The CDB RCS data is organized so that client-devices can easily retrieve the following information from the RCS model (Figure 7-1: Graphical Representation of the 3D Model RCS Shape Data) below:

- The modeling (physical) parameters that were used to generate the RCS polar data.
- The RCS polar representation corresponding to one or more levels of resolution of the RCS polar data.
- The RCS polar representation corresponding to distinct radar mode of operation.
- The RCS polar representation corresponding to a distinct radar model type.

RCS resolution refers to the angular pitch used in gathering RCS data for the model in question. At a given RCS resolution, it is possible to have two or more RCS polar representations due to the fact that the RCS data is computed based on a number of physical modeling properties such as the characteristics of the electromagnetic beam,





its frequency, polarization, amplitude and phase. A simulated sensor operating in a given mode of operation, over a given range of frequencies, will require the RCS data closest to this mode. It will therefore need to use the closest matching Polar Diagram from the RCS model data.

## 7.3 RCS Polar Diagram Data Representation using Shapefile

This section provides a detailed description of the content and format of RCS data for the CDB.

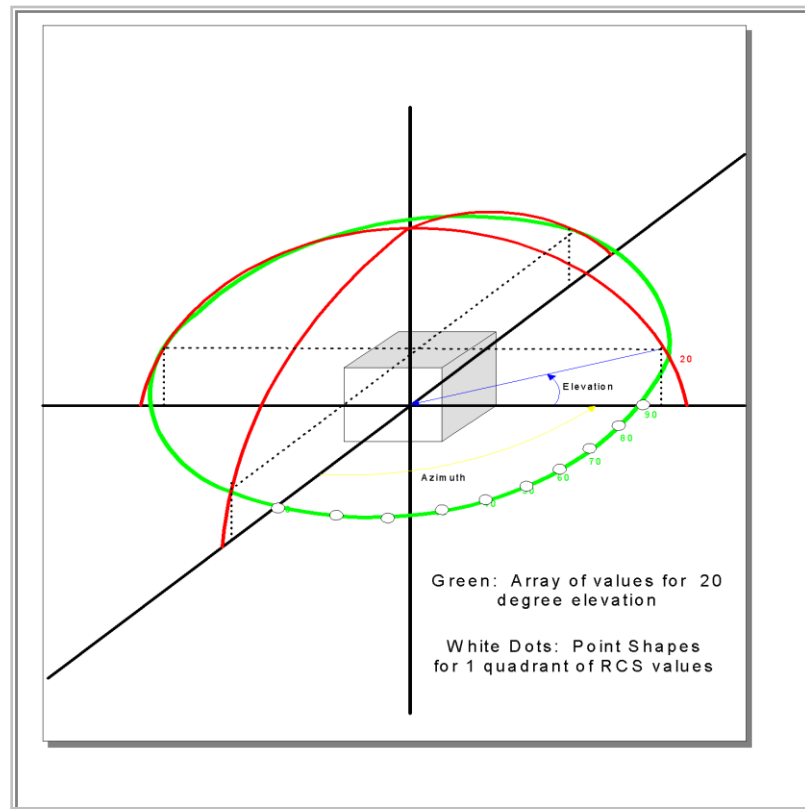
### 7.3.1 Shapefile Internal Data Structure

Within a CDB, the RCS model is stored as a series of ShapeFiles in accordance with the ESRI Shapefile Specification. This section describes the Shapefile internal data structure for the representation of RCS model data. This format provides the required flexibility to create and visualize the RCS data:

- Easy modification of data attributes
- Simple visualization of RCS data in polar form
- Allow irregular steps in azimuth/elevation (X/Y)
- Allow some possibly missing values

RCS data is inherently two-dimensional in nature and is naturally organized as a two-dimensional array of RCS polar values computed at various azimuth and elevation angles from the target. Each element of this array represents the RCS data value over each uniformly distributed azimuth angle and distinct elevation angle.

Therefore, each of such array element can be represented as a “Point” shape, with the azimuth angle value (X) at a given elevation angle (Y), while at the same time storing the associated attributes such as the RCS, Amplitude or Phase data in the instance attribute database (dbf file) associated to the Shapefile. Typical azimuth angles would range between  $-180^{\circ}$  and  $+180^{\circ}$ , where as the elevation angles would cover from  $-90^{\circ}$  to  $+90^{\circ}$ . However, the RCS data set could potentially only cover just a partial range of those angles if data is incomplete for example. This can be visualized in the next diagram, showing RCS values at various azimuth angles corresponding to an elevation angle of  $20^{\circ}$  with respect to the model (cube). Note that the axis conventions follow those described in Section 6.3, Coordinate Systems.



**Figure 7-1: Graphical Representation of the 3D Model RCS Shape Data**

Partial RCS data is permitted, i.e., it is permitted to cover a sub-region of the RCS polar diagram with only points corresponding to known values.

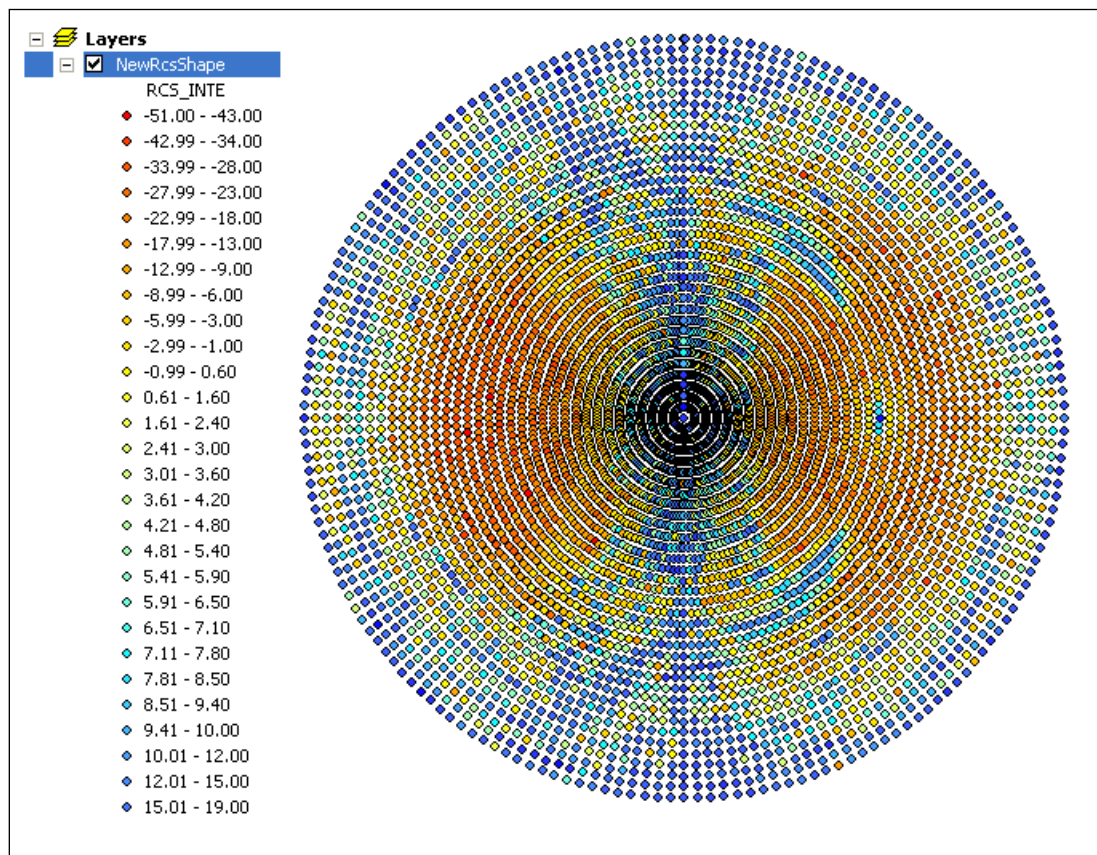
For example, consider an RCS model consisting of data values in  $5^\circ$  elevation increments and  $2^\circ$  azimuth increments covering the entire aspect angle range of the target. The CDB representation would consist of  $(180^\circ/5^\circ)+1 = 37$  sets of  $(360^\circ/2^\circ)+1 = 181$  points (vertices) for a full target aspect coverage; yielding 6697 point shapes with their attribute data. For each of those Shapefile point vertices, the X component represents the azimuth angle (equivalent to longitude) and the Y component represents the elevation angle (equivalent to latitude); the RCS value (and other attributes) being stored in the instance attributes within the DBF file.

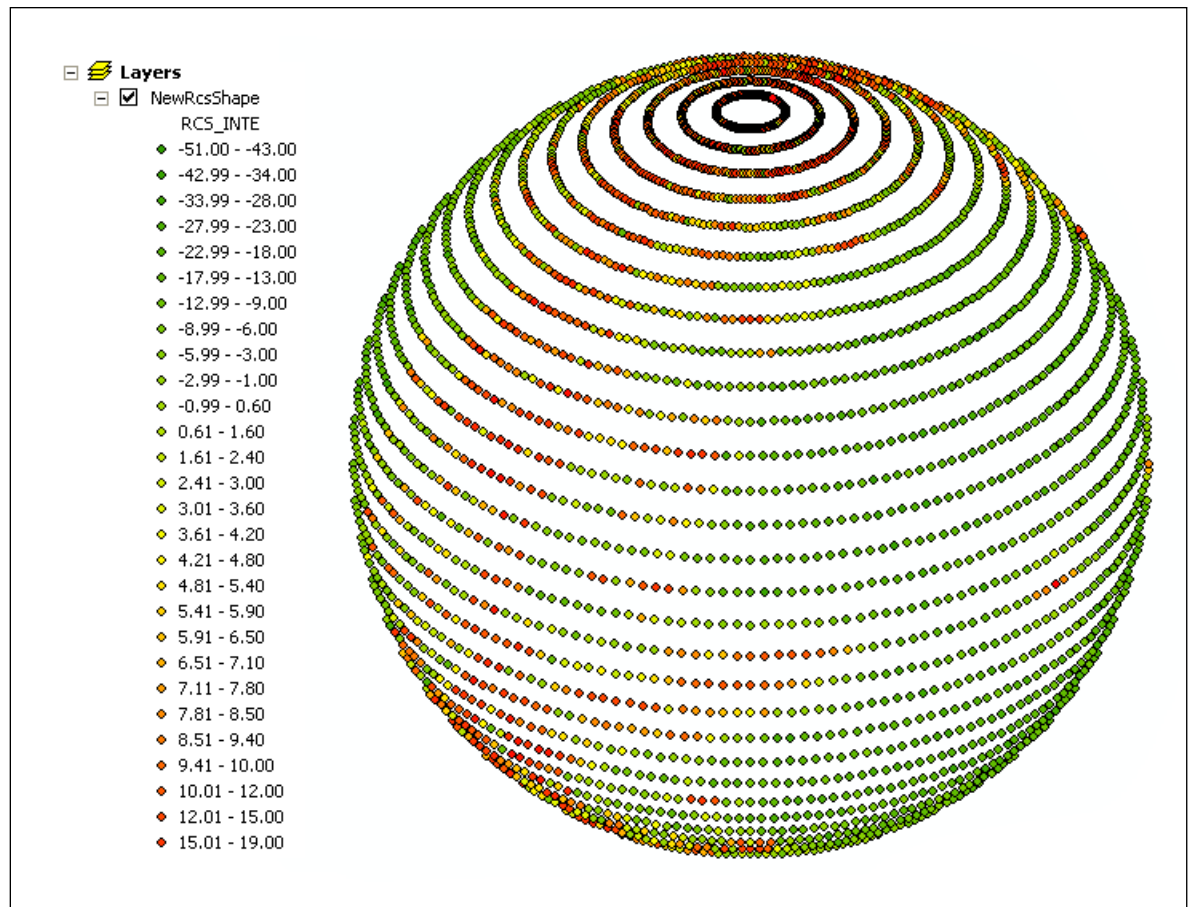
The CDB specification defines eight prescribed values for azimuth and elevation increments. They are referred to herein as ModelSignature Significant Angle. The table below shows the correspondence between the ModelSignature LOD level number and the ModelSignature Significant Angle.

**Table 7-1: ModelSignature Significant Angle per LOD**

ModelSignature LOD level	Significant Angle	Number of values
0	$90^\circ \leq \text{Significant angle}$	Less than 8
1	$45^\circ \leq \text{Significant angle} < 90^\circ$	between 8 and 32
2	$22.5^\circ \leq \text{Significant angle} < 45^\circ$	between 32 and 128
3	$11.25^\circ \leq \text{Significant angle} < 22.5^\circ$	between 128 and 512
4	$5.625^\circ \leq \text{Significant angle} < 11.25^\circ$	between 512 and 2048
5	$2.80^\circ \leq \text{Significant angle} < 5.625^\circ$	between 8192 and 32768
6	$1.40^\circ \leq \text{Significant angle} < 2.80^\circ$	between 32768 and 131072
7	$0.70^\circ \leq \text{Significant angle} < 1.40^\circ$	between 131072 and 524288

Such a data representation would typically produce the following diagram when viewed in 2D (Figure 7-2: Polar Diagram of RCS Data in Planar Representation) and 3D (Figure 7-3: Polar Diagram of RCS Data in Spherical Representation) polar forms (color representing the RCS Intensity attribute):


**Figure 7-2: Polar Diagram of RCS Data in Planar Representation**



**Figure 7-3: Polar Diagram of RCS Data in Spherical Representation**

In addition, specific attributes within the Shapefile are required to specify other characteristics of the RCS data, like EM polarization mode and frequency that were used when characterizing the target's RCS signature. Those are the class-level attributes and are described below.

The data for each distinct RCS representation model requires two different types of attributes; RCS model class attributes and RCS instance attributes.

1. **RCS Model Class-level attribution:** These are attributes that can be shared by all of the RCS model instances of the RCS representation. The attributes and their values are logically re-grouped under a classname that stands for the entire attributes specific to the RCS model. All of the classnames are re-grouped into a model.dbf file referred to as the RCS Class Attribute file for the RCS model. (See Section 7.4.1, Directory Structure) Each row of the model.dbf file corresponds to a different classname. The first column of the file is the classname attribute and acts as the primary key to access subsequent table entries; all other columns correspond to the attributes represented by the classname.



2. **RCS Instance-level attribution:** This is the data that represents a particular instance of the RCS model for a RCS representation. The data is contained in the attribution columns of the model.dbf file that accompanies the RCS's \*.shp file. This \*.dbf file is referred to as the RCS Instance Attribute file of the RCS model. (See Section 7.4.1, Directory Structure) The first column of each row is always the classname attribute. The other columns in a RCS Instance Attribute file are used to describe further the associated shape.

In summary, for a single RCS model in the CDB, the data files consist of:

- One \*.shp main file that provides the geometric aspect (Points) for each data instance of a RCS model.
- Two \*.dbf files (one instance-level on a per RCS Shape basis, and one class-level at the RCS model level) that collectively provide the attribution for all of the possible RCS models of a given RCS Model.
- One \*.shx index file that stores the file offsets and content lengths for each of the records of the main \*.shp file. The only purpose of this file is to provide a simple means for clients to step through the individual records of the \*.shp file (i.e., it contains no CDB modeled data).

#### **7.3.1.1 RCS Model Class-Level Attributes:**

Many attributes within the Shapefile are required to specify the physical modeling parameters corresponding to those used to produce the RCS data; this includes, for instance, the electro-magnetic (EM) polarization mode and the frequency that were used when characterizing the target's RCS signature.

The CDB RCS model representation offers a comprehensive set of class attributes that are described below. It is important to note that these attributes are an elaborate set of fields to indicate in which physical environment the RCS data was computed, and does not necessarily reflect a precise operating mode of a particular radar.

A description of the attribute information follows below. (The reader should keep in mind that the 10-character limitation of attribute names is imposed by the dBASE III+ file format used by the Shapefile .DBF data format)

**Table 7-2: XML Tags for Hot Spots**

<b>Attribute</b>	<b>Format</b>	<b>Description</b>	<b>Values</b>	<b>Units</b>
CLASSNAME	STRING	Unique string identifying the RCS model class attribute characteristics	Uniquely identifiable character string for the class name	String of 32 characters
VERSION	STRING	String representing the version level of the RCS Data	XX.YY.ZZ	String of 8 characters
PROD_DAY	INT	Number representation of the computation day	DD	N/A
PROD_MTH	INT	Number representation of the computation month	MM	N/A



Attribute	Format	Description	Values	Units
PROD_YEA	INT	Number representation of the computation year	YYYY	N/A
CLASS_TYP	INT	Level of Classification	0 – 999 0 : UNKNOWN 1 : UNCLASSIFIED 2 : SECRET 3 : TOP SECRET 4 : DECLASSIFIED 999: OTHER	Enumerated
DAT_SRC_T	INT	Data from which RCS was derived	0 - 999 0 : UNKNOWN 100 : OPENFLIGHT 200 : EMPIRICAL 300 : THIRD-PARTY TOOL 400 : US Air Force 401 : US Army 402 : US Navy 999 : OTHER	Enumerated
RCS_VARI	STRING	Radar Model Variant (e.g., “AN/APG-65”)	7.3.2, Multi-Variant RCS Model Applicability	String of 10 characters
3RD_PARTY	INT	3rd party tool used for RCS Production	0 - 999 0 : UNKNOWN 100 : RADBASE 200 : XPATCH 300 : MATHLAB/SIMULINK 999 : OTHER	Enumerated
POL_TYPE	INT	Polarization Mode of RF emission used to characterize RCS	0- 999 0 : UNKNOWN 1 : LINEAR 2 : CIRCULAR 3 : ELLIPTICAL 4 : SINGLE HH 5 : SINGLE HV 6 : SINGLE VV 7 : SINGLE VH 8 : DUAL HH-HV 9 : DUAL VV-VH 10 : DUAL HH-VV 11 : ALTERNATING HH-HV 12 : ALTERNATING VV-VH 13 : POLARIMETRIC HH 14 : POLARIMETRIC VV 15 : POLARIMETRIC HV 16 : POLARIMETRIC VH 999: OTHER	Enumerated
EX_AMPL	DOUBLE	Transmitted Ex-component amplitude level		INTENS_TY



Attribute	Format	Description	Values	Units
EY_AMPL	DOUBLE	Transmitted Ey-component amplitude level		INTENS_TY
EX_PHASE	DOUBLE	Transmitted Ex-component phase		ANGL_TYP
EY_PHASE	DOUBLE	Transmitted Ey-component phase		ANGL_TYP
EX_FREQ	DOUBLE	Transmitted Ex-component frequency		FREQU_TY
EY_FREQ	DOUBLE	Transmitted Ey-component frequency		FREQU_TY
INTENS_TY	INT	RCS Value units	0 – 999 0 : UNKNOWN 1 : DB 2 : DBSM 3 : VOLTS 4 : SURFACE 5 : M2 999: OTHER	N/A
ANGL_TYP	INT	RCS Angular Value units	0 : UNKNOWN 1 : DEGREES 2 : RADIANS 3 : GRADIANS 4 : STERADIANS	N/A
FREQU_TY	INT	RCS Frequency Value units	0 : UNKNOWN 1 : HERTZ 2 : KILOHERTZ 3 : MEGAHERTZ 4 : GIGAHERTZ 5 : TERAHERTZ 6 : PETAHERTZ	N/A
TGT_TY	INT	Target Mode Value units	0 : UNKNOWN 1 : NORMAL 2 : SLIGHTLY DAMAGED 3 : DAMAGED 4 : DESTROYED	Enumerated
TIME_TY	INT	Time Value units	0 : UNKNOWN 1 : SECONDS 2 : MILLI-SECONDS 3 : MICRO-SECONDS	Enumerated
RF_TY	INT	RF Emission Mode Type	0 : UNKNOWN 1 : CONTINUOUS WAVE 2 : PULSED	Enumerated



Attribute	Format	Description	Values	Units
LENGTH_TY	INT	Length Value units	0 – 999 0 : UNKNOWN 1 : NANOMETER 2 : MICRON 3 : MILLIMETER 4 : CENTIMETER 5 : METER 6 : KILOMETER 999: OTHER	N/A
RF_FREQ	DOUBLE	Frequency of RF emission used to characterize RCS		FREQU_TY
TGT_SS	DOUBLE	Significant size of input Source Model Data		LENGTH_TY
MLOBEGAIN	DOUBLE	Antenna Main Lobe Gain		INTENS_TY
MLOBEBW	DOUBLE	Antenna Main Lobe Bandwidth		ANGL_TYP
SLOBE3DB	DOUBLE	Antenna Side Lobe 3dB Point		ANGL_TYP
RF_PWIDTH	DOUBLE	RF Pulse Width		TIME_TY
RF_PRF	DOUBLE	RF Pulse Repetition Frequency		FREQU_TY
RCS_AVG_I	DOUBLE	RCS Intensity Average (or mean) Value. This represents the arithmetic mean of the RCS table.		INTENS_TY
RCS_AVG_A	DOUBLE	RCS Amplitude Average (or mean) Value. This represents the arithmetic mean of the RCS table.		INTENS_TY
RCS_AVG_P	DOUBLE	RCS Phase Shift Average (or mean) Value. This represents the arithmetic mean of the RCS table.		ANGL_TYP
RCS_NML_I	DOUBLE	Approximated RCS Intensity Value for ‘Normal’ state		INTENS_TY
RCS_NML_A	DOUBLE	Approximated RCS Amplitude Value for ‘Normal’ state		INTENS_TY
RCS_NML_P	DOUBLE	Approximated RCS Phase Shift Value for ‘Normal’ state		ANGL_TY
RCS_SD_I	DOUBLE	Approximated RCS Intensity Value for ‘Slightly Damaged’ state		INTENS_TY



Attribute	Format	Description	Values	Units
RCS_SD_A	DOUBLE	Approximated RCS Amplitude Value for ‘Slightly Damaged’ state		INTENS_TY
RCS_SD_P	DOUBLE	Approximated RCS Phase Shift Value for ‘Slightly Damaged’ state		ANGL_TY
RCS_DMG_I	DOUBLE	Approximated RCS Intensity Value for ‘Damaged’ state		INTENS_TY
RCS_DMG_A	DOUBLE	Approximated RCS Amplitude Value for ‘Damaged’ state		INTENS_TY
RCS_DMG_P	DOUBLE	Approximated RCS Phase Shift Value for ‘Damaged’ state		ANGL_TY
RCS_DST_I	DOUBLE	Approximated RCS Intensity Value for ‘Destroyed’ state		INTENS_TY
RCS_DST_A	DOUBLE	Approximated RCS Amplitude Value for ‘Destroyed’ state		INTENS_TY
RCS_DST_P	DOUBLE	Approximated RCS Phase Shift Value for ‘Destroyed’ state		ANGL_TY
RCS_FLU_I	DOUBLE	RCS Intensity Fluctuation (or Variance); the mean of all squared deviations from the mean for all RCS values.		N/A
RCS_FLU_A	DOUBLE	RCS Amplitude Fluctuation (or Variance); the mean of all squared deviations from the mean for all RCS values.		N/A
RCS_FLU_P	DOUBLE	RCS Phase Fluctuation (or Variance); the mean of all squared deviations from the mean for all RCS values.		N/A
RCS_SCINT	DOUBLE	This value specifies a level of scintillation to be added to the simulated radar signature when model parts are being articulated.	7.3.3, Model’s Articulations Effect on RCS Data	INTENS_TY
RCS_FLASH	DOUBLE	RCS Intensity of Target when viewed directly at 0° (face) or 180° (back) degrees azimuth. This “face” value is sometimes necessary when viewpoint turns around target and gets a “flash” at those specific angles.		INTENS_TY

Attribute	Format	Description	Values	Units
EQ_SPH_RD	DOUBLE	Radius of an approximated equivalent metallic sphere substituting the model		LENGTH_TY
MAX_VAL_I	DOUBLE	RCS Table Max Intensity Value		INTENS_TY
MAX_VAL_A	DOUBLE	RCS Table Max Amplitude Value		INTENS_TY
MAX_VAL_P	DOUBLE	RCS Table Max Phase Shift Value		ANGL_TY
MIN_VAL_I	DOUBLE	RCS Table Min Intensity Value		INTENS_TY
MIN_VAL_A	DOUBLE	RCS Table Min Amplitude Value		INTENS_TY
MIN_VAL_P	DOUBLE	RCS Table Min Phase Shift Value		ANGL_TY
AZ_SSANGL	DOUBLE	Azimuth smallest significant delta angle	Smallest azimuth angle increment found in data	ANGL_TYP
EL_SSANGL	DOUBLE	Elevation smallest delta significant angle	Smallest elevation angle increment found in data	ANGL_TYP
AZ_LSANGL	DOUBLE	Azimuth largest significant delta angle	Smallest azimuth angle increment found in data	ANGL_TYP
EL_LSANGL	DOUBLE	Elevation largest significant delta angle	Smallest elevation angle increment found in data	ANGL_TYP

### 7.3.1.2 RCS Instance-Level Attribute Data

The data for an entire RCS model itself is stored as a series of Point Shapes, each representing the RCS data values with respect to the model's center for the corresponding azimuth and elevation angles as represented by the point X and Y coordinates. The \*.dbf portion of the Shapefile provides the instance attribute information for each of the RCS Point. A description of the attribute information follows below:

**ShapeType** = POINT

Values:

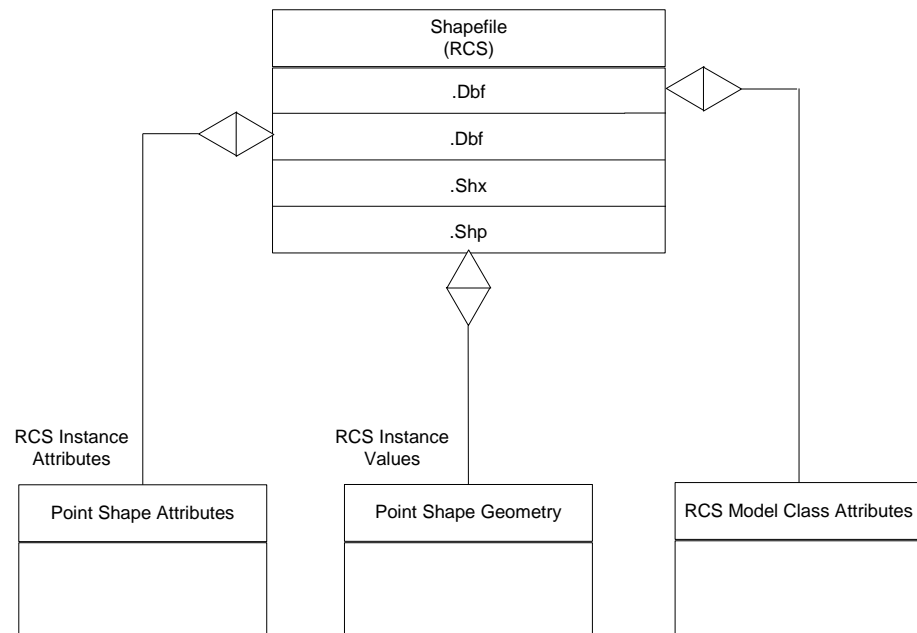
X coordinate is the Azimuth angle of the RCS sample

Y coordinate is the Elevation angle of the RCS sample

**NOTE:** The RCS of the model when viewed at +90° elevation (top view) is significantly different than the one at -90° elevation (bottom view), so there should be  $(180/EL\_STEP)+1$  point values to cover all elevations. The azimuth, which has the same RCS value for +180° and -180° will cover  $(360/AZ\_STEP)$  point values.

**Table 7-3: RCS Instance Attribute Fields**

ATTRIBUTE	TYPE	DESCRIPTION	VALUES	UNITS
CLASSNAME	STRING	Unique string referring to the RCS model class attribute name	String of 32 characters	
RCS_INTE	DOUBLE	RCS Intensity Level		INTEN_TY
RCS_AMPL	DOUBLE	RCS Amplitude Level		INTEN_TY
RCS_PHAS	DOUBLE	RCS Phase		ANGL_TYP



**Figure 7-4: UML Representation of the 3D Model RCS Shapefile Structure**

For a given RCS curve in a Shapefile, an attribute “CLASSNAME” indicates which type of sensor application the curve data is derived for, and under which resolution the data was produced. Therefore, the single Shapefile of the Model can regroup all sensor data pertaining to various RCS signature types and resolutions for a given RCS Model. Consider the next example. The Shapefile format therefore does not preclude the capability to support multiple RCS curves simultaneously for a given model.

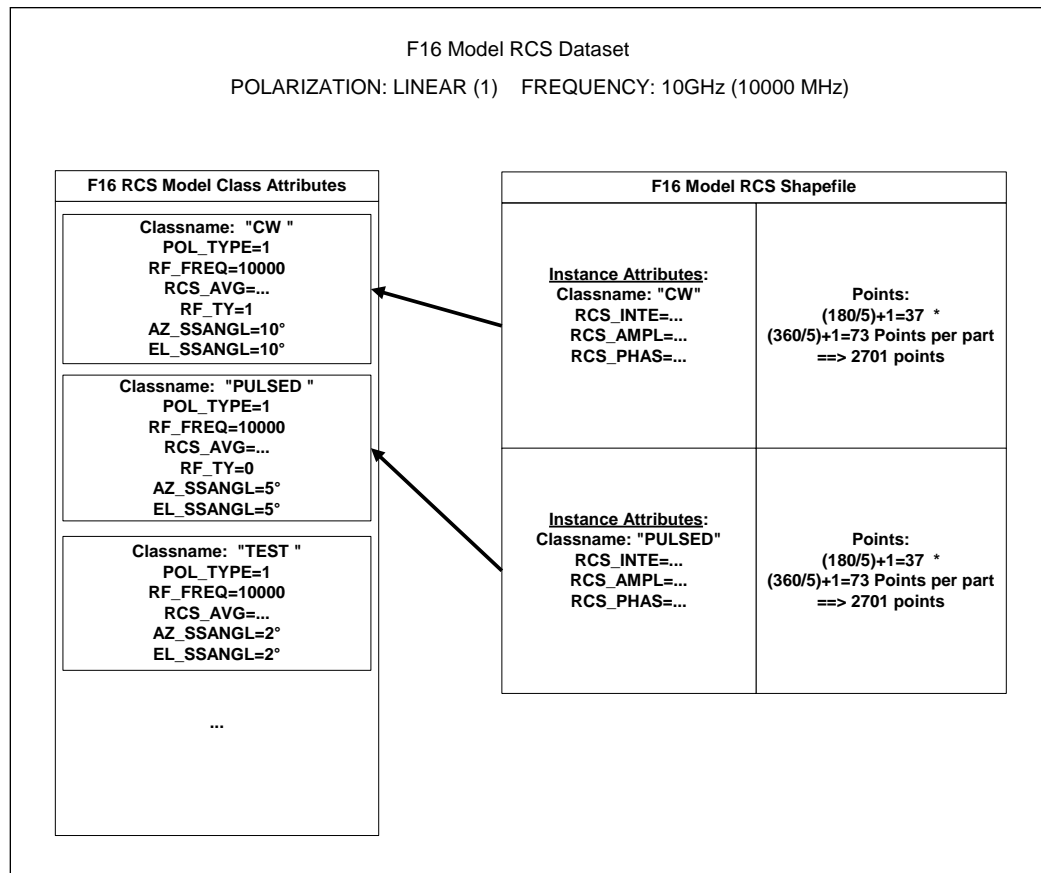


Figure 7-5: Example of RCS Shapefile

### 7.3.2 Multi-Variant RCS Model Applicability

Each variant of the RCS model in the Shapefile has a 10-character string attribute called “RCS\_VARI”. The string may contain the specific Radar model number (and possibly its frequency band L-Band, S-Band, X-Band, Ku-Band) for which this RCS variant applies to. The suggested string convention for this field is as described in reference [22]:

For example: The “AN/APG-65” Radar model name represents a Pulse Doppler X-Band Multi-Mode Radar manufactured by Raytheon (Hughes) and used in F/A-18, AV-8B+ aircraft.

Table 7-4: Radar Model Numbers

AN/APA - Airborne Radar Auxiliary Assemblies

Model Number	Description
AN/APA-1	Indicator Unit (Remote Repeater Scope) used with US Navy ASB radar
AN/APA-2	Radar Antenna Equipment
AN/APA-3	Radar Antenna Equipment



<b>Model Number</b>	<b>Description</b>
AN/APA-4	Radar Alarm Unit
AN/APA-5	Auxiliary Electronic Bombsight Equipment; used in P-2
AN/APA-6	Panoramic Radio Receiving Set; used with AN/APR-9 and AN/APR-14
AN/APA-7	Movie-Camera Photo Set
AN/APA-8	Video Amplifier; used with AN/APS-2
AN/APA-9	ECM Equipment; used in P2V-5
AN/APA-10	Panoramic Radio Receiving Set; used with AN/SPR-2
AN/APA-11	Panoramic Radio Receiving Set (Pulse Analyzer); used in B-52 EW pod, RC-121C, P2V-5, PBM-5S used with AN/APR-9 and AN/APR-14;
AN/APA-12	Sector Scan Antenna Adapter; used with AN/APS-2
AN/APA-13	Component of AN/APS-15
AN/APA-14	Component of AN/APS-15
AN/APA-15	Elevation Stabilizer; used with AN/APS-15
AN/APA-16	Auxiliary Electronic Bombsight Equipment used in PBY-6A
AN/APA-17	250-1000 MHz Broadband Direction Finding Radar (used with search receivers); manufactured by Hoffman Radio Corp., Aviola
AN/APA-19	Bombing Aid
AN/APA-21	Radar Bombing Compensating Unit
AN/APA-23	Automatic Tape Recorder; manufactured by Gamewell; used with AN/APR-1/2/4/5/6
AN/APA-24	50-280 MHz Direction Finding Radar (used with search receivers); manufactured by Heyer Products used in P4M-1Q
AN/APA-25	Radar Direction Finding Antenna Unit
AN/APA-26	S-Band Attenuator
AN/APA-27	Automatic Search & Jam Tuning Adapter
AN/APA-28	Multiple Indicator Equipment (6 displays); used with AN/APQ-13
AN/APA-29	Bombing Altitude Control Unit
AN/APA-33	Multiple Indicator Equipment (4 displays); used with AN/APQ-7
AN/APA-35	Radar Signal Recording Camera Unit
AN/APA-36	Remote Repeater Scope (modified AN/APA-1); used with AN/APQ-13
AN/APA-38	Panoramic Radio Receiving Set; used in PBM-5S
AN/APA-39	Radar Identification Unit
AN/APA-40	Bombing/Navigation System used with AN/APS-15; used in B-17
AN/APA-42	Bombing/Navigation System; used with AN/APS-23; used in XB-48 (see AN/APA-59)
AN/APA-43	Airborne Searchlight Control
AN/APA-44	Ground Position Indicator System; manufactured by Bell Telephone Lab; used with AN/ASB-3 and AN/APS-23/27/31 used in B-45 (together with AN/APS-23 to form AN/APQ-24), RB-66
AN/APA-45	Radar Antenna Tilt Stabilizer Unit
AN/APA-48	Radar Homing Equipment, 140-300 MHz; manufactured by RRL
AN/APA-49	Radar Bombing Ground Position Indicator
AN/APA-50	Low Altitude Rocket Bombing Unit
AN/APA-51	Radar Indicator Unit
AN/APA-52	X-Band TACAN Doppler Navaid; used in F-8, SB-29

Model Number	Description
AN/APA-54	Radar Recorder Group (SHORAN); used in B-57
AN/APA-55	Radar Adapter Unit
AN/APA-56	Radar Display Console; used with AN/APS-45/95; used in EC-121
AN/APA-57	Ground Position Indicator Group; used in AF-2W, P-2, S-2; replaced by AN/ASA-13
AN/APA-58	Ground Position Computer
AN/APA-59	Bombing/Navigation Computer "SRC-1"; manufactured by Sperry; used in B-36, XB-48
AN/APA-60	Autopilot
AN/APA-61	Radar Bombing Navigational Computer
AN/APA-62	Panoramic Receiver
AN/APA-63	Autopilot
AN/APA-64	Radar Signal Analyzer used in P2V-4
AN/APA-66	Radar Monitor
AN/APA-69	Direction Finding Radar Set; used in RB-57D, A-1, C-47, P-2, P-5, S-2, RC-121C, Z-1, ZPK
AN/APA-70	Direction Finder Group; used with AN/APR-9; used in AF-2W, P-2, S-2, TBM-3S
AN/APA-72	Signal Analyzer; used in E-2
AN/APA-74	Pulse Analyzer Group; manufactured by Loral; used in EB-66, A-3, EC-47, P-2, P-5, Z-1, ZPK; replaced AN/APA-11
AN/APA-80	Control & Guidance Monitoring Group; used in AUM-N-2, HSL-1, P-2, P-5, S-2
AN/APA-81	Ground Position Indicator Group; used with AN/APS-20; used in AF-2W, EC-121
AN/APA-82	Direction Finder Group; used in B-52, C-130, C-133, C-135
AN/APA-84	Radar Intercept Targeting Computer; used with APG-37; used in F-86D/K
AN/APA-85	Control-Indicator Group; used with AN/APS-42 used in R6D-1
AN/APA-89	Coder Group; used in A-3, UH-1E
AN/APA-90	Indicator Group; used with AN/APW-11; used in B-57, B-66
AN/APA-91	used with AN/APS-33
AN/APA-92	ECM Set
AN/APA-94	Signal Analyzer
AN/APA-95	Doppler Navigation Computer
AN/APA-106	Bomb Damage Evaluation Group; used with AN/APQ-24; used in B-50D
AN/APA-109	Radar Control; manufactured by Westinghouse
AN/APA-113	used with AN/APS-62
AN/APA-122	Radar Set
AN/APA-125	Radar Display; used with AN/APS-80/82, AN/ASA-47; used in P-2H, P-3A, P-5, E-1
AN/APA-126	Doppler Equipment; used in A-7
AN/APA-127	Sparrow Missile Fire Control System; manufactured by Raytheon; used in F-3, F-4B/C
AN/APA-128	Sparrow Missile Radar Set Group; manufactured by Raytheon; used with AN/AWG-7; used in XF8U-3, F-4
AN/APA-138	Radar Display; used with AN/AWG-7; used in XF8U-3
AN/APA-141	Radar Set; used in B-52G/H
AN/APA-143	Rotodome Antenna Group; manufactured by Dalmo Victor; used with AN/APS-96; used in E-2A/B





<b>Model Number</b>	<b>Description</b>
AN/APA-144	Signal Analyzer Group; used in EA-1F, EC-121M, P-3A
AN/APA-150	Station Keeping System; used in SH-34J
AN/APA-153	Cable Breakout Adapter Set; manufactured by AC Spark Plug; used with AN/APS-104
AN/APA-157	Continuous Wave Illuminator (for AIM-7 targeting); manufactured by Raytheon; used in F-4B/C
AN/APA-159	Radar Set Group; manufactured by Hazeltine; used in EC-121D/H
AN/APA-160	Test Adapter; manufactured by Sperry; used with AN/APN-42
AN/APA-161	Station Keeping System used in ASW helicopters
AN/APA-162	Map Matcher
AN/APA-164	Rotodome; used with AN/APS-111; used in E-2A/B
AN/APA-165	Intercept Computer (for AIM-9 firing); manufactured by Raytheon; used with AN/APQ-109 used in F-4D
AN/APA-167	used with AN/APG-53
AN/APA-170	Radar Set
AN/APA-171	Rotodome Antenna Group; used with AN/APS-120, AN/APX-76; used in E-2C
AN/APA-172	Control Indicator Group; used with AN/APS-120, AN/APX-76; used in E-2C
AN/APA-173	Test Bench

#### AN/APB - Airborne Bombing Radars

<b>Model Number</b>	<b>Description</b>
AN/APB-1	Radar Beacon
AN/APB-2	Bombing Radar; used in B-58

#### AN/APD - Airborne Direction Finding and Surveillance Radars

<b>Model Number</b>	<b>Description</b>
AN/APD-1	Homing Radar; used in TBF/TBM
AN/APD-2	Radar Direction Finding Set; used with AN/APR-1 and AN/APA-48
AN/APD-4	D/E/F-Band Radar Direction Finding System; manufactured by ITT; used in RB-47H, B-52, EB-66C
AN/APD-5	Reconnaissance Radar
AN/APD-7	Radar Surveillance System; manufactured by Westinghouse; used in OV-1D, RA-5C
AN/APD-8	Side-Looking Reconnaissance Radar; manufactured by Westinghouse; proposed for RF-111A
AN/APD-9	Radar Set
AN/APD-10	Side-Looking Reconnaissance and Mapping Radar; used in F-4, RF-4B/C, CP-140; special tests in NC-141, C-130
AN/APD-11	Side-Looking Radar Reconnaissance Set; part of AN/UPD-6; used in RF-4E
AN/APD-12	I/J-band Side-Looking Reconnaissance System; manufactured by Lockheed Martin; part of AN/UPD-8 and AN/UPD-9; used in Israeli RF-4B
AN/APD-13	QUICK LOOK Electronic Intelligence Subsystem; manufactured by Systems & Electronics; used in "Guardrail" RC-12
AN/APD-14	SAROS (SAR for Open Skies) Radar System; manufactured by Sandia; part of AN/UPD-8; used in OC-135

Model Number	Description
AN/APD-501	Maritime Patrol Radar; used in Lancaster (Canada)

#### AN/APG - Airborne Fire Control Radars

Model Number	Description
AN/APG-1	S-Band Intercept Radar used in P-61B
AN/APG-2	S-Band Intercept & Gun Laying Radar used in P-61
AN/APG-3	Tail Gun Laying Radar; manufactured by General Electric used in B-29 and B-36B
AN/APG-4	L-Band Low Altitude Torpedo Release Radar "Sniffer" used in TBM
AN/APG-5	S-Band Gun Laying/Range-Finding Radar used in B-17, B-24 and F-86A (AN/APG-5C)
AN/APG-6	L-Band Low Altitude Bomb Release Radar "Super Sniffer" (improved AN/APG-4)
AN/APG-7	Glide Bomb Control Radar "SRB" (Seeking Radar Bomb)
AN/APG-8	S-Band Turret Fire Control Radar used in B-29B
AN/APG-9	L-Band Low Altitude Bomb Release Radar (improved AN/APG-6)
AN/APG-10	Weapons System Radar
AN/APG-11	L-Band Toss Bombing Radar
AN/APG-12	L-Band Low Altitude Bomb Release Radar (improved AN/APG-9)
AN/APG-13	S-Band Nose Gun Laying Radar "Falcon"; manufactured by General Electric used with 75mm nose gun of B-25H
AN/APG-14	S-Band Gun Sight Radar used in B-29
AN/APG-15	S-Band Tail Gun Radar used in B-29B, PB4Y
AN/APG-16	X-Band Gun Laying Radar (modification of AN/APG-2) used in B-32, XB-48
AN/APG-17	S-Band Low Altitude Bomb Release Radar (improved AN/APG-4)
AN/APG-18	X-Band Turret Control Radar (improved AN/APG-5); manufactured by Martin used with "S-4" gunfight
AN/APG-19	X-Band Fire Control Radar; manufactured by Martin (improved AN/APG-8 and -18)
AN/APG-20	S-Band Low Altitude Bomb Release Radar (improved AN/APG-6)
AN/APG-21	Ground-Ranging Radar
AN/APG-22	X-Band Gun Sight Radar; manufactured by Raytheon used with Mk.18/23 Lead-Computing Gun Sights
AN/APG-23	Weapons System Radar used in B-36A
AN/APG-24	Weapons System Radar used in B-36B
AN/APG-25	X-Band Gun Tracking Radar used in F-100
AN/APG-26	Weapons System Tracking Radar; manufactured by Westinghouse used in F3D
AN/APG-27	Tail Gun Radar used in XB-46 and XB-48
AN/APG-28	Intercept Radar (modified AN/APG-1) used in F-82F
AN/APG-29	Night/All-Weather Fighter Fire-Control Radar (for Type D-1 Fire-Control System)
AN/APG-30	X-Band Fire Control Radar; manufactured by Sperry used in B-45, B-57, F-4E, F-8A, F-84E, F-86A (final blocks only), F-86E/F, F-100, FJ-2, F2H-2
AN/APG-31	Gun Laying Radar; manufactured by Raytheon used in B-57
AN/APG-32	X-Band Tail Turret Autotrack Radar; manufactured by General Electric used in B-36D/F, B-47E
AN/APG-33	X-Band Fire Control Radar; manufactured by Hughes used in TB-25K, F-94A/B, F-89A



<b>Model Number</b>	<b>Description</b>
AN/APG-34	Computing Radar Gunfight used in F-104C
AN/APG-35	Radar used in F3D
AN/APG-36	Search Radar used in F2H-2N, F-86D (replaced by AN/APG-37)
AN/APG-37	Search Radar; manufactured by Hughes used in F-86D/K/L, F2H-4
AN/APG-39	Gun Laying Radar used in B-47E
AN/APG-40	Fire Control Radar; manufactured by Hughes used in TB-25M, F-94C, F-89D, CF-100 (Canada)
AN/APG-41	Tail Gun Radar (twin radomes); manufactured by General Electric used in B-36H
AN/APG-43	Continuous Wave Interception Radar; manufactured by Raytheon
AN/APG-45	Fire-Control Radar (miniaturized AN/APG-30); manufactured by General Electric; intended for patrol aircraft gun turrets
AN/APG-46	Fire-Control Radar; tested in A-6A
AN/APG-48	Airborne Fire-Control System Mk.22
AN/APG-50	Intercept Radar used in F-4
AN/APG-51	Intercept Radar; manufactured by Hughes used in F3H-2, F3D
AN/APG-53	Weapons System Radar; manufactured by Stewart-Warner used in A-4
AN/APG-55	Pulse Doppler Intercept Radar; manufactured by Westinghouse
AN/APG-56	Fire Control Radar (similar to AN/APG-30) used in F-86 (only Australian models with A-4 gun sight)
AN/APG-57	Fire-Control Radar; manufactured by Gould
AN/APG-59	Pulse-Doppler Gunnery Radar; manufactured by Westinghouse; part of AN/AWG-10 used in F-4J
AN/APG-60	Doppler Radar; part of AN/AWG-11 used in F-4K
AN/APG-61	Fire-Control Radar; part of AN/AWG-12 used in F-4M
AN/APG-63	Pulse Doppler X-Band Fire Control Radar (AN/APG-63(V)2 is an AESA variant); manufactured by Raytheon (Hughes) used in F-15A/B/C/D/H/K
AN/APG-64	Fire-Control Radar (development of AN/APG-63); not produced
AN/APG-65	Pulse Doppler X-Band Multi-Mode Radar; manufactured by Raytheon (Hughes) used in F/A-18A/B, F-4 ICE/Peace Ikarus 2000, AV-8B+ (upgraded)
AN/APG-66	Pulse Doppler X-Band Multi-Mode Radar; manufactured by Northrop Grumman (Westinghouse) used in F-16A/B, F-4EJ (Japan), Hawk 200 (UK)
AN/APG-67	Pulse Doppler X-Band Multi-Mode Radar; manufactured by Lockheed Martin (General Electric) (Model G-200) used in F-20, A-50 (Korea), F-5-2000 (Taiwan), Ching Kuo (Taiwan)
AN/APG-68	Pulse Doppler X-Band Multi-Mode Radar (improved AN/APG-66); manufactured by Northrop Grumman (Westinghouse) used in F-16C/D-30/40/50
AN/APG-69	Radar Set; manufactured by Emerson used in F-5E, AV-8?
AN/APG-70	Pulse Doppler X-Band Multi-Mode Radar (upgrade of AN/APG-63); manufactured by Raytheon (Hughes) used in F-15C/D/E
AN/APG-71	Pulse Doppler X-Band Multi-Mode Radar; manufactured by Raytheon (Hughes) used in F-14D
AN/APG-73	Pulse Doppler X-Band Multi-Mode Radar (upgrade of AN/APG-65); manufactured by Raytheon (Hughes) used in F/A-18C/D/E/F
AN/APG-74	Pod-mounted Radar System; manufactured by Northrop Grumman (Norden)
AN/APG-76	Pulse Doppler Ku-Band Multi-Mode Radar; manufactured by Northrop Grumman (Norden) used in F-4E (Israel); tested in pod with F-16, S-3B

Model Number	Description
AN/APG-77	Pulse Doppler X-Band AESA (Active Electronically Scanned Array) Multi-Mode Radar; manufactured by Northrop Grumman/Raytheon used in F/A-22A
AN/APG-78	Fire Control Radar "Longbow"; manufactured by Northrop Grumman & Lockheed Martin used on mast in AH-64D, RAH-66, underwing on AH-1W/Z
AN/APG-79	AESA (Active Electronically Scanned Array) Multi-Mode Radar (based on AN/APG-73); manufactured by Raytheon used in F/A-18E/F/G as replacement for AN/APG-73
AN/APG-80	"Agile Beam Radar" AESA (Active Electronically Scanned Array) Multi-Mode Radar (based on AN/APG-68); manufactured by Northrop Grumman; intended for F-16E/F
AN/APG-81	AESA (Active Electronically Scanned Array) Radar planned for F-35
AN/APG-501	X-Band Ranging Radar used in F-86
AN/APG-T1	Radar Training Set for AN/APG-1

#### AN/APN - Airborne Navigation Radars

Model Number	Description
AN/APN-1	Radio Altimeter (improved AN/ARN-1) used in P-61, C-119, B-32, C-121, H-19, P-5, AF-2W, AD-5, F2H-2/2N/2P, F3D, F6F-5N, F9F, XF10F-1, P2V-4, PB4Y-2, PBM-5S, PBX-6A, R5C-1, R5D-2, R6D-1, SB2C-5, TBM-3S
AN/APN-2	"Rebecca" Radio Beacon used with AN/PPN-1, AN/TPN-2
AN/APN-3	SHORAN used with AN/CPN-2 used in B-45A
AN/APN-4	LORAN; manufactured by Philco used in B-29, B-32, C-47, C-54, C-117, C-121, P2V-4, PBM-5S, PBX-6A, PB4Y-2, R4Q-1, R6D-1
AN/APN-5	Radar Beacon Navigation Aid used in F-86
AN/APN-6	S-Band Beacon used with AN/PPN-10, AN/PPN-11
AN/APN-7	LORAN S-Band Beacon used with AN/APS-2
AN/APN-8	Radar Beacon
AN/APN-9	LORAN; manufactured by RCA used in B-29, B-32, RC-121, C-97 replaced AN/APN-4
AN/APN-10	"Rebecca" Interrogation Set
AN/APN-11	X-Band Beacon used with AN/APS-3/4/6/10/15/31/33 used in B-47, KC-97, XS-1
AN/APN-12	Rendezvous Radar (or 160-230 MHz "Rebecca" Interrogator) used in B-47, C-97
AN/APN-13	S-Band Beacon (improved AN/APN-7)
AN/APN-14	Navigation Aid
AN/APN-15	Low Level Altimeter Set; manufactured by Sperry used in B-52, CH-3C
AN/APN-16	Radar Beacon
AN/APN-18	Radar Beacon
AN/APN-19	"Rosebud" S-Band Beacon used in F-82D
AN/APN-20	Radar Beacon
AN/APN-21	Radar Beacon
AN/APN-22	Radar Altimeter; manufactured by Electronic Assistance Corp used in A-3, B-66, C-119, RC-121, C-130, RF-101C, OV-1, AD-5, P2V-5, R6D-1
AN/APN-23	Active Seeker used in KAY-1(XSAM-N-4)
AN/APN-24	Navigation Set
AN/APN-25	Doppler Navigator; manufactured by GPI



<b>Model Number</b>	<b>Description</b>
AN/APN-26	SG-Band (VHF) Beacon
AN/APN-29	SG-Band (VHF) Beacon
AN/APN-30	Radar Beacon
AN/APN-33	S-Band Beacon; replaced AN/APN-7 used in XSSM-N-8
AN/APN-34	Distance Measuring Radar used in C-97C, R6D-1
AN/APN-35	Radar Beacon
AN/APN-36	Radar Beacon
AN/APN-37	Radar Beacon
AN/APN-38	Radar Beacon
AN/APN-39	Radar Beacon
AN/APN-40	Radar Beacon
AN/APN-41	Missile Beacon for LTV-N-2 replaced AN/APN-33
AN/APN-42	Radar Altimeter used in WC-130, WB-47E, B-52
AN/APN-45	Tracking Radar Beacon used in DC-130A
AN/APN-46	Radar Beacon
AN/APN-47	Radar Beacon
AN/APN-48	Radar Beacon
AN/APN-49	Radar Beacon
AN/APN-50	Navigation Radar; manufactured by Sperry
AN/APN-52	Radar Set
AN/APN-54	Radar Beacon
AN/APN-55	Radar Beacon (for missiles)
AN/APN-56	Navigation Radar; manufactured by Gould
AN/APN-57	Ground Position Indicator
AN/APN-58	Navigation Radar; manufactured by Sperry
AN/APN-59	Search & Weather Radar; manufactured by Sperry used in C-130, C-135, B-57, C-133, C-141, KC-97
AN/APN-60	S-Band Beacon used in B-52
AN/APN-61	Radar Beacon used in XF-85
AN/APN-63	S-Band (Receive)/L-Band (Transmit) Beacon; manufactured by Melpar
AN/APN-66	Doppler Navigation Radar used in SM-62, B-47
AN/APN-67	Doppler Set used in P6M-1, NC-121 "Project Magnet", USN helicopters; tested in P-2
AN/APN-68	IFF Beacon used with AN/APX-6
AN/APN-69	X-Band Rendezvous Beacon used in B-47, B-52, C-97, RB-57D, KC-135 used with AN/APN-59
AN/APN-70	LORAN; manufactured by Dayton Aviation Radio & Equip Corp used in B-50, C-54, C-119, C-121, RC-121D, C-130, C-135, P-2, P-3A, T-29C/D, Z-1, R6D-1
AN/APN-71	Flare-Out Unit
AN/APN-75	Rendezvous Radar used in B-47
AN/APN-76	Rendezvous Radar; manufactured by Olympic used in KC-97, B-47B/E
AN/APN-77	Doppler Set used in SZ-1B, USN helicopters
AN/APN-78	Doppler Set used in helicopters
AN/APN-79	Doppler Set manufactured by Teledyne Ryan used in helicopters

Model Number	Description
AN/APN-81	Doppler Set used in RB/WB-66, WB-50, C-130, KC-135
AN/APN-82	Doppler Navigation Radar (combination of AN/APN-81 and AN/ASN-6) used in EB/RB/WB-66, KC-135
AN/APN-84	SHORAN Set; manufactured by Hazeltine used in RC-130A
AN/APN-85	Navigation Radar; manufactured by Hazeltine
AN/APN-89	Doppler Set; part of AN/ASQ-38 used in B-52E/G/H
AN/APN-90	Doppler Set
AN/APN-91	Tracking Beacon used in BQM-34C
AN/APN-92	Navigation Radar
AN/APN-96	Doppler Set
AN/APN-97	Doppler Set; manufactured by Ryan used in UH-2A, SH-3, SH-34J
AN/APN-99	Doppler Navigation Set (combination of AN/APN-81 and AN/ASN-7) used in B-52, AC-130A, KC-135
AN/APN-100	Radar Altimeter; manufactured by Litton used in CH-47A
AN/APN-101	Airborne Radar used in RF-4C, F-4E (possible confusion with AN/ARN-101)
AN/APN-102	Doppler Set; manufactured by GPI used in RB-47, WB-47E, RB-57F, WB-57F, F-100C/F, RF-101
AN/APN-103	Navigational Computer System
AN/APN-105	All-Weather Doppler Navigation System; manufactured by LFE used in F-105B/D, T-39B
AN/APN-107	Navigation Radar used in RB-57D
AN/APN-108	Doppler Set (derivative of AN/APN-89 with gyro components from AN/APN-81) used in B-52E
AN/APN-109	Altimeter; manufactured by Honeywell
AN/APN-110	Doppler Navigation Set used in B-58, F-100D/F, RF-101
AN/APN-113	Doppler Radar; part of AN/ASQ-42 used in B-58
AN/APN-114	Automatic Landing System used with AN/GSN-5; tested in TF-102
AN/APN-115	Navigation Radar; manufactured by General Electric
AN/APN-116	Doppler Set
AN/APN-117	Low-Level Radar Altimeter (in combination with AN/APN-22); manufactured by Electronic Assistance Corp used in A-6A, P-2, S-2, SH-3A, H-13H, CH-47A, HH-52, CH-53A
AN/APN-118	Doppler Navigation Set
AN/APN-119	Doppler Set
AN/APN-120	Radar Altimeter; planned for A-5, A-6A, but not produced
AN/APN-122	Doppler Navigation Set used in S-2, A-2, A-3, A-4, A-6, RA-5C, C-47, C-54, EC-121, E-2, TF-8, P-2, P-3, P-5
AN/APN-126	Doppler Set
AN/APN-127	Collision Warning System
AN/APN-128	Navigation Radar; manufactured by Teledyne used in C-130
AN/APN-129	Doppler Navigation System; manufactured by Teledyne used in OV-1A/B
AN/APN-130	Doppler Radar; manufactured by Teledyne Ryan used in UH-2, SH-3, SH-34J, CH-53D, Z-1
AN/APN-131	Doppler Navigation Radar used in F-105, T-39B, TF-8A
AN/APN-132	X-Band Beacon; manufactured by Motorola used in BQM-34A, QF-9G





Model Number	Description
AN/APN-133	High-Altitude Radar Altimeter (upgraded SCR-728) used in C-130, C-135
AN/APN-134	Ku-Band Beacon; manufactured by Bendix used in KC-135
AN/APN-135	X-Band Beacon (for in-flight refueling); manufactured by Bendix used in B-58
AN/APN-136	Ku-Band Beacon (for in-flight refueling); manufactured by Bendix used in B-58
AN/APN-140	Radar Altimeter
AN/APN-141	Low Altitude Radar Altimeter; manufactured by Bendix used in A/TA-4, A-6, A-7, C-2, C-130, C-141, E-2C, F-4, F-8, F-104, F-105, P-3, S-2, T-39, SH-3
AN/APN-142	Navigation Radar used in F-4C
AN/APN-144	Doppler Navigation Radar used in EC-121, VC-137
AN/APN-145	LORAN C Set used in RC-135D
AN/APN-146	Radar Altimeter
AN/APN-147	Doppler Navigation System; manufactured by Canadian Marconi used in AC-119, C-124C, C-130, WC-130B/E, RC-135A, WC-135B, C-135F, C-141
AN/APN-148	Doppler Navigation Radar used in F-105D/F
AN/APN-149	Terrain Avoidance Radar used in TF-8
AN/APN-150	Radar Altimeter used in CH-3C, B-52, C-130, EC-130E, C-135
AN/APN-151	LORAN C Receiver; manufactured by ITT used in RC-135B, C-141A, H-3
AN/APN-152	LORAN C Receiver
AN/APN-153	Doppler Navigation Radar used in A-6, A-4, EA-6A/B, A-7, C-130G, E-2, P-3A, S-2E
AN/APN-154	X-Band Beacon Augmenter (Tracking Beacon); manufactured by Motorola used with AN/TPB-1, AN/TPQ-10 used in A-4, A-7, F-14, A-6, AH-1T, H-46, CH-53
AN/APN-155	Low Altitude Radar Altimeter; manufactured by Stewart-Warner used in F-4
AN/APN-157	LORAN C Receiver; manufactured by ITT used in C-130, RC-135B, C-141, P-3C, EP-3E
AN/APN-158	Weather Radar; manufactured by Collins used in HC-123B, U-8F, U-21A, CV-2
AN/APN-159	Radar Altimeter; manufactured by Stewart-Warner used in RF-4
AN/APN-161	High-Resolution Mapping Radar; manufactured by Sperry used in C-130
AN/APN-162	manufactured by Canadian Marconi
AN/APN-163	Doppler Navigation System
AN/APN-165	Terrain-Following/Ground-Mapping Radar; manufactured by Texas Instruments used in OV-1
AN/APN-167	Radar Altimeter; manufactured by Honeywell used in F/FB-111A
AN/APN-168	Doppler Radar; manufactured by Canadian Marconi used with AN/AYA-3 used in CH-53A, OV-1
AN/APN-169	Station-Keeping Radar; manufactured by Sierra Research used in C-130, C-141
AN/APN-170	Terrain Following Radar; manufactured by General Dynamics; tested in A-4C, B-52, B-58
AN/APN-171	Radar Altimeter; manufactured by Honeywell used in C-130, E-2C, SH-2F, SH-3H, OH-6A, CH-46, CH-53
AN/APN-172	Doppler Set; manufactured by Marconi used with AN/ASN-73 used in HH-53C, CH-53G
AN/APN-174	Station-Keeping Subsystem; manufactured by Teledyne used in CH-46, CH-53
AN/APN-175	Doppler Radar used in C-130, CH-3B, HH-3E, MH-53
AN/APN-176	Radar Altimeter; manufactured by Texas Instruments used in FB-111A



Model Number	Description
AN/APN-177	Doppler Altimeter
AN/APN-178	Navigation Radar; manufactured by Sierra used in C-130
AN/APN-179	Doppler Navigation Radar; manufactured by Bendix used in EC-47
AN/APN-180	LORAN A Automatic Tracking Receiver used with AN/AYN-1 used in HH-3F
AN/APN-181	LORAN C/D Receiver
AN/APN-182	Doppler Radar Navigation System; manufactured by Ryan used with AN/AYK-2 used in SH-3H, CH-46, HH-46A/D, SH-2D, UH-2C, RH-53
AN/APN-184	Radar Altimeter; manufactured by Bendix used in C-130, Hawker P-1127 (UK)
AN/APN-185	Doppler Navigation Radar; manufactured by Singer-Kearfott used in FB-111A, A-7D, B-1A
AN/APN-186	Doppler System; tested in A-6 ILAAS (AN/ASQ-116)
AN/APN-187	Doppler Navigation Radar; manufactured by Singer-Kearfott used in P-3
AN/APN-189	Doppler Navigation Radar; manufactured by Marconi used in F-111D
AN/APN-190	Doppler Radar; manufactured by Singer-Kearfott used in A-7, AC-130E, F-111
AN/APN-191	Radar Altimeter used in A-7D
AN/APN-192	Short-Pulse Radar Altimeter; manufactured by Teledyne used in CH-47
AN/APN-193	Doppler Velocity Sensor; manufactured by Ryan
AN/APN-194	Radar Altimeter; manufactured by Honeywell used in F-14, A-6E, AH-1W, HH-60H, EA-6B, AV-8B, C-2A, P-3C, EP-3E, F/A-18, SH-60B/F, T-45A, TA-4J, TC-130G, S-3, A-4, A-7, A-10, B-1, TC-4C, QF-4, BQM-8D/F, MQM-8G, BQM-34S, AQM-34U, RGM/UGM-109B
AN/APN-195	Nose-Mounted Radar; manufactured by Collins used in SH-3D, HH-3E
AN/APN-196	Doppler Radar used in F-105
AN/APN-197	STATE Airborne Station; manufactured by Honeywell used with AN/TPN-21, AN/UPN-33; tested in C-123, C-131, T-39, CH-3
AN/APN-198	Radar Altimeter; manufactured by Honeywell used in F-104G, AV-8, Sea King (UK), Lynx (UK)
AN/APN-199	LORAN Receiver; manufactured by Collins used in C-5A
AN/APN-200	Doppler Velocity Sensor; manufactured by Teledyne used in B-1, E-3, S-3
AN/APN-201	Radar Altimeter; manufactured by Hoffman Electronics used in S-3
AN/APN-202	Radar Beacon; manufactured by Motorola used with AN/SPN-46 ACLS (Automatic Carrier Landing System) used in AV-8B, F/A-18, S-3, C-2, P-3C
AN/APN-203	Radar Altimeter; manufactured by Stewart-Warner used in T-43A
AN/APN-205	Doppler Radar; manufactured by Teledyne used in SH-2, SH-60B
AN/APN-206	Doppler Set used in B-1A
AN/APN-208	Doppler Navigation Radar; manufactured by Marconi used in HH-53H, Bell 412
AN/APN-209	Radar Altimeter; manufactured by Honeywell/Stewart-Warner used in AH-1F, UH-1V, CH-47D, OH-58C/D, H-60, T-43A
AN/APN-210	Doppler Set; manufactured by Singer used in CH-53G
AN/APN-211	Navigation Radar; manufactured by Teledyne-Ryan used in helicopters
AN/APN-213	Doppler Velocity Sensor; manufactured by Litton (Teledyne-Ryan) used in E-3; tested in KC-135
AN/APN-214	Radar Altimeter
AN/APN-215	Weather & Search Radar; manufactured by Bendix/King used in RU-38A, U-21, C-130



<b>Model Number</b>	<b>Description</b>
AN/APN-217	Doppler Radar Navigation Sensor; manufactured by Litton (Teledyne-Ryan) used in AH-1W, UH-1N, SH-2G, SH-3D, HH-3F, CH-46, CH-53E, MH-53E, RH-53D, HH-60H/J, SH-60B/F/J, V-22
AN/APN-218	Doppler Radar Navigation System; manufactured by Litton (Teledyne-Ryan) used in B-1B, B-52G/H, KC-135, C-130, F-111G
AN/APN-220	Doppler Radar; manufactured by Teledyne-Ryan
AN/APN-221	Doppler Radar (derived from AN/APN-208); manufactured by Marconi used in C-141, HH-53H, MH-53J
AN/APN-222	Radar Altimeter; manufactured by Honeywell used in C-130, E-6A
AN/APN-224	Radar Altimeter; manufactured by Honeywell used in B-52G/H, B-1B
AN/APN-227	Doppler Radar used in P-3C
AN/APN-230	Doppler Navigation Radar (improved AN/APN-218) used in B-1B
AN/APN-231	Radar Navigation System; manufactured by Teledyne-Ryan used in EA-6A
AN/APN-232	CARA (Combined Altitude Radar Altimeter); manufactured by Gould used in C-5, C-17, C-130, OC-135, C-141, F-16
AN/APN-233	Doppler Navigation Radar (developed from AN/APN-220); manufactured by Teledyne-Ryan used in C-2, OV-10D, CH-47, S-2, Alpha Jet (Germany), DHC-5
AN/APN-234	Weather and SAR Radar (Model RDR-1400C; improved AN/APN-215); manufactured by Telephonics (originally by Bendix/King) used in P-3, C-2
AN/APN-235	Doppler Navigation Set (development of AN/APN-221) used in HH-60A
AN/APN-236	Doppler Radar System; manufactured by Teledyne
AN/APN-237	Ku-Band Terrain-Following Radar; manufactured by Texas-Instruments; part of AN/AAQ-13
AN/APN-238	
AN/APN-239	Weather and SAR Radar (Model RDR-1400C, similar to AN/APN-234); manufactured by Telephonics (originally by Bendix/King) used in HH-60G, MH-60G
AN/APN-240	Station-Keeping System; manufactured by Sierra Research; replaced AN/APN-169
AN/APN-241	Weather & Navigation Radar; manufactured by Northrop Grumman (Westinghouse) used in C-130H/J, C-27J, HS-748 (Australia)
AN/APN-242	Weather & Navigation Radar; manufactured by Sperry; replacement for AN/APN-59
AN/APN-243	Station-Keeping Equipment; manufactured by Sierra Technologies used in C-17, C-130J
AN/APN-244	E-TCAS (Enhanced Traffic Alert Collision Avoidance System); manufactured by Honeywell (AlliedSignal) used in C-130E/H/J
AN/APN-245	Radar Beacon used with ACLS (Automatic Carrier Landing System) AN/SPN-46 used in F/A-18
AN/APN-501	Doppler Radar used in C-141(?)
AN/APN-503	Doppler Radar used in CP-121 (Canada)
AN/APN-509	Radar Altimeter
AN/APN-510	Doppler Navigation System used in CP-140 (Canada)
AN/APN-511	Radar Altimeter
AN/APN-512	Radar Altimeter used in CC-130E/H (Canada)
AN/APN-513	Doppler Radar Navigation Set used in CH-124A (Canada)
AN/APN-T6	Radar Interpretation Trainer
AN/APN-T8	Doppler System Trainer used with C-5

Model Number	Description
AN/APN-T10	Radar Trainer used with C-5

#### AN/APQ - Airborne Multipurpose/Special Radars

Model Number	Description
AN/APQ-1	Radar Jammer RT-26
AN/APQ-2	450-750 MHz High Power Barrage Jamming Transmitter "Rug"; manufactured by General Motors (Delco Div.) used in PB4Y-2
AN/APQ-3	S-Band Radar Receiver; later redesignated AN/APR-5
AN/APQ-4	Panoramic Radar Receiver; later redesignated AN/APR-6
AN/APQ-5	Low Level Radar Bombsight; manufactured by Western Electric used with AN/APS-2/3/15 used in B-24, B-25, B-32, PBJ, PBM
AN/APQ-7	X-BAND Search & Bombing Radar "Eagle Mk.1"; manufactured by Western Electric used in B-17, B-24, B-25J, B-29, B-32
AN/APQ-8	Deception Radar "Spoofers"
AN/APQ-9	475-585 MHz High Power Barrage Jamming Transmitter "Carpet III"; manufactured by General Motors (Delco Div.)
AN/APQ-10	X-Band High Altitude Bombing Radar "Eagle Mk.2"; manufactured by Western Electric used in B-29
AN/APQ-11	Torpedo Launching Radar (formerly SCR-626)
AN/APQ-12	Torpedo & Bombing Radar (formerly SCR-631)
AN/APQ-13	X-Band Bombing Radar "Mickey" (British equivalent is H2X); manufactured by Western Electric used in B-29, B-32
AN/APQ-14	Radar "Moth-1"
AN/APQ-15	88-162 MHz Radar Spoofing Transmitter "Moonshine"; manufactured by RRL
AN/APQ-16	Radar Bombing Aid
AN/APQ-17	Radar Jammer
AN/APQ-19	S-Band Search & Bombing Radar
AN/APQ-20	S-Band Radar Jammer; manufactured by RRL, Delmont Radio; uses AN/APA-41, AN/APR-10, AN/APT-10
AN/APQ-21	Countermeasures Set; similar to AN/SPT-7
AN/APQ-22	Radar System
AN/APQ-23	X-Band High Altitude Bombing Radar used in B-29
AN/APQ-24	K-1 Radar Navigation & Bombing System used in B-36B, B-45A, B-50, B-66B
AN/APQ-27	Radar Jamming System; uses AN/APT-16 (2x), AN/APR-9
AN/APQ-29	Radar Relay Set
AN/APQ-31	Bombing & Navigation Radar
AN/APQ-32	RT-119 Radar Jammer
AN/APQ-33	Countermeasures Set used in AC-119K
AN/APQ-34	K-Band Bombing Radar; manufactured by Western Electric
AN/APQ-35	X-Band Search, Fire Control & Tail-Warning Radar (components are AN/APS-21, AN/APS-28, AN/APG-26); manufactured by Westinghouse used in F3D, F2H, F3H
AN/APQ-36	Search & Acquisition Radar; manufactured by Westinghouse used in F3D-2M, F7U-3M
AN/APQ-39	Weather Radar(?) used in B-52D



Model Number	Description
AN/APQ-41	X-Band Search & Intercept Radar (improved AN/APQ-35); manufactured by Westinghouse used in F3D-2, F2H-3
AN/APQ-43	Multipurpose Radar; designated AI22 in UK used in Javelin FAW.2/6 (UK)
AN/APQ-46	Radar Set; proposed for F3D-3
AN/APQ-50	X-Band Fighter Interceptor Radar; manufactured by Westinghouse used in F-4, F3H, F4D; planned for F12F
AN/APQ-51	X-Band Missile Radar; manufactured by Sperry used in F3H, F7U
AN/APQ-54	Chronograph Set (projectile velocity measuring equipment)
AN/APQ-55	K-Band Side-Looking Radar used in RF-4C
AN/APQ-56	Side-Looking, Real-Aperture Radar; manufactured by Westinghouse used in RB-57D, RB-47
AN/APQ-57	Millimeter-Wavelength Navigation Radar
AN/APQ-58	Millimeter-Wavelength Navigation Radar
AN/APQ-59	Side-Looking Airborne Radar; manufactured by Westinghouse
AN/APQ-60	Missile Illumination Radar; manufactured by Raytheon
AN/APQ-62	Side-Looking Radar
AN/APQ-63	Radar
AN/APQ-64	Radar used in F5D with AAM-N-3/AIM-7B Sparrow II missile
AN/APQ-65	Interception Radar used in Aquilon 203 (French-built D.H. Vampire)
AN/APQ-67	Interception Radar; manufactured by Raytheon
AN/APQ-68	HIRAN used in RC-130A
AN/APQ-69	Experimental SLAR Pod for B-58; manufactured by Hughes
AN/APQ-70	Millimeter-Wavelength Navigation Radar
AN/APQ-72	X-Band Intercept Radar; manufactured by Westinghouse used in F-4 (replaced AN/APQ-50); tested in F3D
AN/APQ-73	Side-Looking Radar; planned for RS-70
AN/APQ-74	X-Band Missile Control Radar used with AN/APA-138, AN/APX-20, AN/APN-22
AN/APQ-81	Doppler Navigation Radar; manufactured by Northrop used in SM-62; planned for F6D and tested in A-3
AN/APQ-83	Fire-Control Radar; manufactured by Magnavox used in F-8D
AN/APQ-84	Radar used in F-8
AN/APQ-86	K-Band Side-Looking Surveillance & Mapping Radar; manufactured by Texas Instruments used in RL-23D, RU-8D
AN/APQ-88	Tracking Radar; manufactured by Naval Avionics used in A-6 (replaced by AN/APQ-112)
AN/APQ-89	Terrain Following Radar; tested in T-2
AN/APQ-92	Ku-Band Search Radar; manufactured by Norden used in A-6, EA-6B, AP-2H
AN/APQ-93	Synthetic-Aperture Ground-Mapping Radar
AN/APQ-94	Radar Set; manufactured by Magnavox used in F-8D/E, T-39D
AN/APQ-95	Collision Avoidance Radar used in helicopters
AN/APQ-96	Radar Set used in OV-10A
AN/APQ-97	K-Band Side-Looking Imaging Radar; manufactured by Westinghouse; tested in OV-1A, YEA-3, DC-6
AN/APQ-99	J-Band Forward-Looking Multipurpose Radar; manufactured by Texas Instruments used in A-7A, RF-4B/C, RF-101

Model Number	Description
AN/APQ-100	Search & Mapping Radar; manufactured by Westinghouse used in F-4C, RF-101
AN/APQ-101	Terrain Following Radar; manufactured by Texas Instruments
AN/APQ-102	Side-Looking Mapping Radar; manufactured by Goodyear used in RB-57, RF-4B/C
AN/APQ-103	Search Radar; manufactured by Norden used in EA-6A, A-6B
AN/APQ-104	Radar Set; manufactured by Magnavox (similar to AN/APQ-94 used in F-8E(FN)
AN/APQ-105	Distance Integrating Set used in RC-135
AN/APQ-107	Radar Altimeter Warning System used with AN/APN-117 used in CH-47A, P-3A/C, EP-3E, S-2, SH-3H
AN/APQ-108	Mapping Radar (SAR?); developed by Conductron used in SR-71
AN/APQ-109	Fire Control & Search Radar; manufactured by Westinghouse used in F-4C/D/E
AN/APQ-110	Ku-Band Terrain Following Radar; manufactured by Texas Instruments used in RF-4C, F/FB-111
AN/APQ-111	X-Band Altimeter-Recorder used with AN/ASQ-92 in KC-135
AN/APQ-112	Tracking Radar; manufactured by Norden used in A-6
AN/APQ-113	Ku-Band Search & Attack Radar; manufactured by General Electric used in F-111, F-5E
AN/APQ-114	Ku-Band Attack Radar; manufactured by General Electric used in F/FB-111A, F-4, F-5E
AN/APQ-115	Terrain Following Radar; manufactured by Texas Instruments used in "Combat Talon" C-130E, A-7A, F-111, RF-4C
AN/APQ-116	Terrain Following Radar; manufactured by Texas Instruments used in A-7A/B/C, C-130
AN/APQ-117	Terrain-Following & Attack Radar (development of AN/APQ-109) used in F-4D/E
AN/APQ-118	Terrain Following Radar; manufactured by Norden used in MH-53H, AH-56A
AN/APQ-119	Ground Mapping & Interception Radar (modified AN/APQ-113); manufactured by General Electric used in F-111A/D
AN/APQ-120	X-Band Fire Control Radar; manufactured by Westinghouse used in F-4D/E/F/G
AN/APQ-122	X-Band Multimode (Terrain Mapping/Target Locating/Navigation/Weather) Radar; manufactured by Raytheon (Texas Instruments) used in MC-130E/H, KC-135A, RC-135A/C, T-43A, C-130, E-4B
AN/APQ-123	used in F-111
AN/APQ-124	Navigation & Fire-Control Radar; manufactured by Magnavox used in F-8J
AN/APQ-125	Doppler Ranging Radar; manufactured by Magnavox used in F-8J
AN/APQ-126	J-Band Terrain Following Radar; manufactured by Raytheon (Texas Instruments) used in A-7D/E, T-39D, AC-130E, CH-53
AN/APQ-127	Forward Looking Radar; manufactured by Sperry-Rand used with AN/ASQ-116; tested in A-6
AN/APQ-128	J-Band Terrain Following Radar; manufactured by Sperry used in A-7D/E, F-111C/D
AN/APQ-129	Search Radar used in EA-6B
AN/APQ-130	Attack Radar; manufactured by Rockwell Autonetics used in F-111D
AN/APQ-131	Target Acquisition Radar; manufactured by Texas Instruments used in OP-2E
AN/APQ-133	X-Band Side Looking Tracking Radar; manufactured by Motorola used in AC-119K, C-130, AC-130
AN/APQ-134	Ku-Band Terrain Following Radar; manufactured by Texas Instruments used in F/FB-111A





<b>Model Number</b>	<b>Description</b>
AN/APQ-135	Sink-Rate Radar System used in A-4, F-4, F-8, C-130, CH-47
AN/APQ-136	Search Radar; manufactured by Texas Instruments used in AC-119K, AC-130A
AN/APQ-137	Moving Target Indicator Radar; manufactured by Emerson used in AH-1G
AN/APQ-138	Radar Set
AN/APQ-139	Ku-Band Multi-Mode Radar; manufactured by Texas Instruments used in B-57G
AN/APQ-140	J-Band Multi-Mode Scan Radar; manufactured by Raytheon (E-Systems); planned for B-1A; tested in KC-135
AN/APQ-141	Terrain Following Radar; manufactured by Norden used in AH-56, HH-53 Pave Low
AN/APQ-142	Surveillance Radar "Quick Look I" used in RV-1C
AN/APQ-144	Ku-Band Attack Radar (improved AN/APQ-113); manufactured by General Electric used in F-111F, FB-111A
AN/APQ-145	Mapping & Ranging Radar; manufactured by Stewart-Warner used in A-4E/F/N/S/SU
AN/APQ-146	Ku-Band Terrain Following Radar; manufactured by Texas Instruments used in F-111F
AN/APQ-148	J-Band Navigation & Attack Radar; manufactured by Norden used in A-6E, TC-4C
AN/APQ-149	Navigation & Fire Control Radar used in F-8
AN/APQ-150	Beacon Tracking Radar used in AC-130E/H
AN/APQ-152	Topographical Mapping Radar; manufactured by Goodyear used in RC-130
AN/APQ-153	I-Band Fire Control Radar; manufactured by System & Electronics Inc. (Emerson Electric) used in F-5E/F
AN/APQ-154	Terrain-Following Radar; manufactured by Texas Instruments used in HH-53C
AN/APQ-155	Strategic Radar; manufactured by Northrop Grumman (Norden) used with AN/ASQ-176 used in B-52H
AN/APQ-156	J-Band Navigation & Attack Radar (improved AN/APQ-148); manufactured by Northrop Grumman (Norden) used in A-6E, TC-4C
AN/APQ-157	I-Band Fire Control Radar (modified AN/APQ-153); manufactured by System & Electronics Inc. (Emerson Electric) used in F-5E/F
AN/APQ-158	Terrain Following Radar (improved AN/APQ-126); manufactured by Raytheon used in MH-53J
AN/APQ-159	I/J-Band Multipurpose Radar (improved AN/APQ-153); manufactured by System & Electronics Inc. (Emerson Electric) used in F-5E/F
AN/APQ-160	Attack Radar used in EF-111A
AN/APQ-161	Attack Radar; manufactured by General Electric used in F-111F
AN/APQ-162	Forward Looking Radar (development of AN/APQ-99?) used in RF-4C
AN/APQ-163	Forward Looking Radar; manufactured by General Electric used in B-1
AN/APQ-164	Pulse Doppler I-Band Multimode Radar; manufactured by Northrop Grumman (Westinghouse) used in B-1B
AN/APQ-165	Attack Radar; manufactured by Texas Instruments used in F-111C
AN/APQ-166	Strategic Radar used in B-52G/H
AN/APQ-167	Radar Set (development of AN/APQ-159); developed by ESCO used in T-47
AN/APQ-168	Multi-Mode Radar; manufactured by Raytheon (Texas Instruments) used in HH-60D, MH-60K; proposed for V-22
AN/APQ-169	J-Band Attack Radar (upgraded AN/APQ-165); manufactured by Lockheed Martin (General Electric) used in F-111C
AN/APQ-170	Terrain Following Radar; manufactured by System & Electronics used in MC-130H

Model Number	Description
AN/APQ-171	Attack & Terrain Following Radar (improved AN/APQ-128/146); manufactured by Raytheon (Texas Instruments) used in F-111C/F
AN/APQ-172	J-Band Terrain Following Radar (upgraded AN/APQ-99); manufactured by Raytheon (Texas Instruments) used in RF-4C
AN/APQ-173	Radar Set; manufactured by Norden; proposed for A-6F
AN/APQ-174	Multi-Mode Radar; manufactured by Raytheon used in MV-22, MH-60K, MH-47E; MH-53
AN/APQ-175	X/Ku-Band Multi-Mode Radar; manufactured by Systems & Electronics Inc. used in C-130E
AN/APQ-178	used in E-2C (developmental item only?)
AN/APQ-179	Control Indicator Set (Display System) used in E-2C
AN/APQ-180	Pulse Doppler Attack Radar (modification of AN/APG-70); manufactured by Raytheon (Hughes) used in AC-130U
AN/APQ-181	Synthetic Aperture J-Band Multi-Mode Radar; manufactured by Raytheon (Hughes) used in B-2A
AN/APQ-183	Multi-Mode Radar; manufactured by Northrop Grumman (Westinghouse); was planned for cancelled A-12A, a derivative was used in RQ-3A
AN/APQ-186	Multi-Mode Radar (improved AN/APQ-174); manufactured by Raytheon used in CV-22
AN/APQ-501	Radar Altitude Warning System used in CP-140?; replaced AN/APQ-107
AN/APQ-T1	Trainer for Aircraft Gun Laying Radar
AN/APQ-T10	Bombing/Navigation Simulator used with B-52D
AN/APQ-T11	Bombing/Navigation Radar Trainer used with B-47, B-52, B-58
AN/APQ-T12	Bombing/Navigation Radar Trainer used with B-47, B-52, KC-97, KC-135

#### AN/APS - Airborne Search & Detection Radars

Model Number	Description
AN/APS-1	X-Band Radar (conflicting references to purpose: either Mapping/Bombing or Tail-Warning)
AN/APS-2	S-Band Search Radar & Beacon used with AN/APQ-5 used in PBJ-1 (if w/o AN/APS-3), PBM-5S, PB4Y-2
AN/APS-3	X-Band Search & Bombing Radar used in PBJ-1, OA-10, PBY-6A, TBM-1D/3E, P-82F
AN/APS-4	X-Band Intercept Radar; manufactured by Western Electric used in C-47, C-117, P-38J, P-82D/F/H, AD, XBT2C-1, F4U-4E, F6F-3E/5E, SB2C-5, SBF-4E, TBF-3, TBM-3S; tested in JRB; British designation is AI Mk XV
AN/APS-5	Intercept Radar (development of AN/APS-4); manufactured by Western Electric used in F4U-4N
AN/APS-6	Intercept Radar (development of US Navy AIA radar); manufactured by Sperry used in P-38M, F2H-2N, F-82D, F6F-3N/5N, F7F-4N, F8F-1N/2N, F4U-4N/5N; tested in SNB-1
AN/APS-7	Search Radar (or Tail-Warning Radar?); manufactured by Westinghouse
AN/APS-8	Conflicting data! I have references for: ASW Search Radar used in P-2E wingtip pod; and Tracking Radar for KDB-1(MQM-39 used in AJ-2P
AN/APS-9	Search Radar used in FR-1N
AN/APS-10	X-Band Search Radar





<b>Model Number</b>	<b>Description</b>
AN/APS-11	Tail Warning Radar
AN/APS-12	Fire Control Radar
AN/APS-13	Tail Warning Radar used in P-38L, P-47D, P-51, P-61, P-63, P-82D, PBJ
AN/APS-14	Gun Laying Radar used in B-17, B-24
AN/APS-15	X-Band Bombing & Navigation Radar "Mickey" (equivalent to British "H2S"); manufactured by Philco used in B-29, PBM-3C/5/5E, B-17, B-24, PB4Y-2, PV-2, PBM-5S
AN/APS-16	L-Band Bomber Tail Warning Radar
AN/APS-17	S-Band Bomber Tail Warning Radar
AN/APS-18	Early Warning Radar (another source has this as Drone Radar used with AN/ARR-9)
AN/APS-19	X-Band Search & Intercept Radar; manufactured by Sperry used in AD-4N/5/6, F2H-2N, F4U-5N, F7F-4N, F8F-1N
AN/APS-20	S-Band Search & Early-Warning Radar; manufactured by Hazeltine/General Electric used with AN/ARW-35 and AN/ART-28 used in TBM-3W, WV-2, PB-1W, ZPG-2W(EZ-1), AF-2W, HR2S-1W, P-2, WB-29, RC-121C, Gannet (UK), Shackleton (UK)
AN/APS-21	Search Radar; manufactured by Westinghouse; part of AN/APQ-35 used in F3D, Meteor NF (UK)
AN/APS-23	Search Radar; manufactured by Western Electric; part of AN/ASB-3 used in B-36, B-45C, B-47E, XB-48, B-50, B-52, C-130, C-135
AN/APS-24	Radar Set used with System 416L
AN/APS-25	Search Radar used in XF10F-1
AN/APS-27	Search Radar used in B-52, RB-66, C-130, C-135
AN/APS-28	Search Radar used in F3D
AN/APS-29	Search Radar
AN/APS-30	Search Radar used in AF-2S
AN/APS-31	Search Radar; manufactured by Westinghouse used in P5M, PBM-3, A-1, P-2, U-16, AF-2S
AN/APS-32	Search Radar used in TBM-3
AN/APS-33	X-Band Search Radar used in S-2F, P4M, P2V-6, ZPG-1W, ZPK
AN/APS-34	Search Radar (similar to AN/APS-33)
AN/APS-35	Search & IFF Radar; manufactured by Philco?
AN/APS-37	Search Radar
AN/APS-38	Search Radar used in S-2
AN/APS-42	Weather Radar; manufactured by Bendix used in C-54, C-97, C-118, C-119, C-121, C-124, C-130, C-131
AN/APS-44	Search Radar used in PB4Y-2, P-5
AN/APS-45	Height-Finding Radar; manufactured by Texas Instruments used in WV-2(EC-121)
AN/APS-46	Interception Radar used in F2H-2N
AN/APS-48	Unattended Radar
AN/APS-49	Rapid Scan Search Radar; manufactured by Hazeltine used for ASW
AN/APS-50	Search Radar; planned for F11F-1, but not used
AN/APS-54	Tail-Warning Radar System; manufactured by ITT used in B-47B/E, B-52, B-57, EB-66B, F-101A/C, F-105D, "EF-101B" (Canada)

Model Number	Description
AN/APS-57	X-Band Search & Intercept Radar; manufactured by Western Electric used in Venom NF.3 (UK; designated AI Mk 21)
AN/APS-59	Search Radar used in CP-109 (Canada)
AN/APS-60	High-Altitude Mapping Radar used in NRB-57A
AN/APS-61	Monopulse Radar
AN/APS-62	Height-Finding Radar used in ZPG-2W/3W
AN/APS-63	Radar Set used in B-66, T-29, F-4C (tests?)
AN/APS-64	Search Radar used in WB-47E, B-52, RB-66B/C
AN/APS-67	Search Radar Set; manufactured by Magnavox used in F-8B, S-2
AN/APS-69	Height-Finding Radar used in ZPG-2W, P-2
AN/APS-70	Early-Warning Radar; manufactured by General Electric used in P2V-6, EC-121, EZ-1C
AN/APS-73	X-Band Synthetic Aperture Radar; manufactured by Goodyear used in experimental pod for B-58; tested in C-97, C-135, RF-4C; ground-component in AN/GSQ-28
AN/APS-75	"SABRE" High-Resolution X-Band Side-Looking Radar; manufactured by General Electric; under consideration for B-70
AN/APS-76	Search Radar used in EC-121
AN/APS-80	Maritime Surveillance Radar; manufactured by Texas Instruments used in E-1B, P-3A/B, NP-3D, P5M-2
AN/APS-81	Search Radar used in B-52
AN/APS-82	Early Warning/Aircraft Direction Radar; manufactured by Hazeltine used in EC-121L, E-1B, E-2; tested in SH-3G
AN/APS-84	Tracking Radar used with QB-47
AN/APS-85	Side-Looking Surveillance Radar; manufactured by Motorola used with RL-23D, RU-8D
AN/APS-87	Early Warning Radar (development of AN/APS-82)
AN/APS-88	Search Radar; manufactured by Texas Instruments used in HU-16B, S-2
AN/APS-91	Early Warning Radar used in E-2
AN/APS-92	Radar Warning Receiver used in F-105D
AN/APS-94	Side-Looking Airborne Surveillance & Mapping Radar; manufactured by Motorola used in OV-1B/D, P-2, P-3, EA-6A, UH-1 ALARM, B-26
AN/APS-95	Search & Warning Radar; manufactured by Hazeltine used in EC/RC-121
AN/APS-96	Air Surveillance Radar; manufactured by General Electric used in E-2A/B
AN/APS-103	Height Finding Radar used in EC/RC-121
AN/APS-104	Bombing/Navigation Radar System; part of AN/ASQ-48 used in B-52C/D
AN/APS-105	Radar Homing & Warning System; manufactured by Dalmo-Victor used in B-52
AN/APS-107	Radar Homing & Warning System; manufactured by Bendix used for targeting AGM-78 used in A-7D, F-105G, F-111A, F-4D; improved version tested in F-4E
AN/APS-108	Search Radar; manufactured by Motorola/Raytheon used in B-52D
AN/APS-109	Radar Homing & Warning System; manufactured by Dalmo-Victor used in F-111A/D/E/F, FB-111A
AN/APS-111	UHF Air Surveillance Radar (modified AN/APS-96); manufactured by Lockheed Martin (General Electric) used in E-2A
AN/APS-112	Early Warning Radar AWACS (development of AN/APS-59)



Model Number	Description
AN/APS-113	Weather Radar; manufactured by Bendix; manufactured by Bendix used in EC-47, UH-1
AN/APS-115	X-Band Sea Surveillance/ASW Radar; manufactured by Raytheon (Texas Instruments) used in P-3C, SH-2D
AN/APS-116	X-Band Sea Surveillance/ASW Radar; manufactured by Raytheon (Texas Instruments) used in EP-3E, S-3A, SH-3, CP-140 (Canada; Canadian version called AN/APS-506), P-3C (Australia); proposed for cancelled U-2EPX
AN/APS-117	TIAS (Target Identification & Acquisition System) for AGM-45 used in some A-4
AN/APS-118	TIAS (Target Identification & Acquisition System) for AGM-78; manufactured by IBM used in A-6B (Mod 1)
AN/APS-119	Weather Avoidance Search Radar used in HC-130B
AN/APS-120	Air Surveillance Radar; manufactured by General Electric used in E-2C
AN/APS-121	Radar Set
AN/APS-122	Search Radar used in YSH-2E
AN/APS-123	Search Radar used in S-2D
AN/APS-124	Sea Surveillance/ASW Radar; manufactured by Raytheon used in SH-60B, YSH-2E; tested in SH-3
AN/APS-125	Pulse Doppler UHF Air Surveillance Radar; manufactured by Lockheed Martin (General Electric) used in E-2C, EC-130V; replaced AN/APS-120
AN/APS-126	Surface Search Radar used in P-3
AN/APS-127	Raytheon Sea Surveillance Radar; manufactured by Raytheon used in HU-25A/B, Gulfstream III (Denmark)
AN/APS-128	Sea Surveillance Radar; manufactured by Telephonics used in E-9A, P-95 (Brazil), D.3B (Spain)
AN/APS-130	Multimode Search Radar (derivative of AN/APG-156); manufactured by Northrop Grumman (Norden) used in EA-6B
AN/APS-131	Sideways Looking Sea Surveillance Radar; manufactured by Motorola used in HU-25B, C-130
AN/APS-133	X-Band Multifunction Radar; manufactured by Allied Signal (Model RDR-1F) used in EA-6A, C-5, KC-10, C-17, EC-24A, VC-25, C-130, C-141, E-3, E-4, E-6, E-8
AN/APS-134	Multimode Search Radar; manufactured by Raytheon (Texas Instruments) used in P-3B, EP-3E, HC-130H, CP-140A (Canada; Canadian version called AN/APS-507), Atlantique (Germany/France), P-3K (New Zealand), Fokker 50 Mk 2 (Singapore), CN-235MPA (Brunei), P-3C (South Korea)
AN/APS-135	Side-Looking Airborne Surveillance Radar; manufactured by Motorola used in HC-130H
AN/APS-136	I-Band MTI Radar; planned for EH-60C
AN/APS-137	Pulse Doppler X-Band Sea Surveillance/ASW Radar; manufactured by Raytheon used in: - AN/APS-137(V)1: A-6E, S-3B - AN/APS-137(V)2: PHM2 Hydrofoil - AN/APS-137(V)3: P-3C - AN/APS-137(V)4: HC-130H - AN/APS-137(V)5: P-3C - AN/APS-137(V)6: ES-3A - AN/APS-137(V)? : EP-3E
AN/APS-138	Pulse Doppler UHF Air Surveillance Radar (upgraded AN/APS-125); manufactured by Lockheed Martin (General Electric) used in E-2C; planned for P-3AEW

Model Number	Description
AN/APS-139	Pulse Doppler UHF Air Surveillance Radar (upgraded AN/APS-138); manufactured by Lockheed Martin used in E-2C(Grp.I)
AN/APS-140	I/J-Band Multimode Surveillance Radar (US version of AN/APS-504); manufactured by Litton Canada
AN/APS-141	I/J-Band Multimode Surveillance Radar (US version of AN/APS-504(V)3); manufactured by Litton Canada
AN/APS-143	X-Band Sea Surveillance Radar "Ocean Eye"; manufactured by Telephonics used in E-9A, S-2E, HU-25, SH-60, SH-2G (Australia, New Zealand), and in aerostats
AN/APS-144	Pulse Doppler Ku-Band Land Surveillance Radar; manufactured by AIL used in EO-5, RQ-5A(BQM-155A); tested in C-27, UH-60A
AN/APS-145	Pulse Doppler UHF Air Surveillance Radar (upgraded AN/APS-139); manufactured by Lockheed Martin used in E-2C(Grp.II), EC-130V
AN/APS-146	manufactured by Northrop Grumman; intended for EA-6B
AN/APS-147	Multi-Mode Surveillance Radar; manufactured by Telephonics used in MH-60R
AN/APS-148	"SeaVue" Lightweight Multi-Platform Sea/Land Surveillance Radar; manufactured by Raytheon
AN/APS-149	Pod-Mounted Surveillance Radar used on P-3C (to provide targeting coordinates of mobile targets for the AGM-84H)
AN/APS-150	Sea Surveillance Radar; modified AN/APS-115 (or AN/APS-137?) for use with C-130; probably used on HC-130H
AN/APS-503	I-Band Multimode Surveillance Radar; manufactured by Litton Canada used in CH-124
AN/APS-504	I/J-Band Multimode Surveillance Radar (improved AN/APS-503); manufactured by Litton Canada used in EC/RC-26D (AN/APS-504(V)5), CP-121
AN/APS-505	Beacon-Equipped Multimode Radar
AN/APS-506	Maritime Surveillance Radar (Canadian version of AN/APS-116); manufactured by Raytheon (Texas Instruments) used in CP-140
AN/APS-507	Maritime Surveillance Radar (Canadian version of AN/APS-134); manufactured by Raytheon (Texas Instruments) used in CP-140A
AN/APS-509	Search Radar used in S-2T
AN/APS-T1	Air-to-Surface Vessel Radar Trainer
AN/APS-T2	Air-to-Surface Vessel Radar Trainer

#### AN/APY - Airborne Surveillance Radars

Model Number	Description
AN/APY-1	Pulse Doppler S-Band Air & Sea Surveillance Radar (AWACS); manufactured by Northrop Grumman used in E-3
AN/APY-2	Pulse Doppler S-Band Air & Sea Surveillance Radar (AWACS); manufactured by Northrop Grumman used in E-3
AN/APY-3	Sideways Looking Air-to-Ground Surveillance Radar (JSTARS); manufactured by Northrop Grumman used in E-8
AN/APY-6	Multi-Mode High Resolution Surveillance Radar; manufactured by Northrop Grumman; tested in NP-3C
AN/APY-7	Sideways Looking Air-to-Ground Surveillance Radar (improved AN/APY-3) used in E-8



Model Number	Description
AN/APY-8	"Lynx" SAR/GMTI (Synthetic Aperture Radar/Ground Moving Target Indicator); manufactured by General Atomics; tested in C-12, U-21 and others; planned for use in MQ-9A
AN/APY-9	UHF Air Surveillance Radar; manufactured by Lockheed Martin used in E-2D
AN/APY-10	Maritime Surveillance Radar; manufactured by Raytheon used in P-8A
AN/APY-12	"Phoenix" SAR (Synthetic Aperture Radar)
AN/APY-T1	RMTS (Radar Maintenance Training Set); part of E-3 AWACS MTS (Maintenance Training System)
AN/APY-T2	ARMTS (Advanced Radar Maintenance Training Set); part of E-3 AWACS MTS (Maintenance Training System)

### 7.3.3 Model's Articulations Effect on RCS Data

Most man-made models (aircraft, tanks, trucks, etc.) have parts that can be articulated (flaps, turrets, rotating antennae, landing gears, etc). It is impractical to pre-compute and store within the CDB an RCS model for every possible position of those articulated parts taken individually. Instead, a CDB RCS model attribute provides the means to store an overall RCS variation effect, or otherwise called "scintillation effect". The scintillation effect value is added to the RCS at run-time during movement of any of such articulated parts of the model. It is a parameter in the Shapefile called "RCS\_SCINT" and this attribute can be used by the radar client-devices at runtime to provide a correlated (but approximated) variation level of the model RCS while any of its parts are articulated.

For example, for a tank in the process of rotating its turret, the radar simulation client would take its overall RCS (based on aspect angles) and add the scintillation factor on the end-result RCS value to slightly alter the RCS to introduce the turning turret effect while the part is moving. While this adds an approximation factor on the RCS, it provides a coherent and correlated variation level to all clients using the RCS data set layer. The "RCS\_SCINT" is therefore the value that represents a scaling factor of RCS noise that would be superimposed while the part is being articulated.

## Chapter 8

### 8 Glossary

	Description
<b>A</b>	
<b>Aliasing</b>	Spatial and/or temporal image defects or artifacts in a raster image. They are due to interaction between the discrete sampling of the raster format and the spatial and temporal frequencies inherent in the computed image of edges, surfaces and point features. Manifestation of spatial aliasing includes edge stair-stepping and crawling, scintillation of narrow surfaces, break-up of long, narrow surfaces and positional or angular motion of scene edges in discrete steps. Temporal aliasing includes double image and loss of dynamic image integrity due to the human eye's ability to dynamically track individual fields at certain angular rates of motion.
<b>Ambient Illumination</b>	See Illumination, Ambient.
<b>Anti-aliasing</b>	Active image processing techniques that reduce the perception of the aliasing phenomena.
<b>Area, Background</b>	An area of unlimited size (up to whole earth) modeled in accordance to the following criteria: (a) Elevation grid post spacing, 30 m. (b) Terrain imagery resolution, 1 m. (c) Contains all objects of height greater than, 33 m (100 ft).
<b>Area, Corridor</b>	A 10 nm wide corridor joining two target areas modeled in accordance to the following criteria: (a) Elevation grid post spacing, 30 m. (b) Terrain imagery resolution, 1 m. (c) Contains all objects of height greater than, 33 m (100 ft).
<b>Area, HLZ</b>	A 0.1 nm x 0.1 nm area inside the target (see "Target" glossary entry) modeled in accordance to the following criteria: (a) Elevation grid post spacing, 1 m. (b) Terrain imagery resolution, 0.1 m. (c) Contains all objects of height greater than, 0.15 m (0.5 ft).
<b>Areal Feature</b>	See Feature, Areal



	Description
<b>Area, Target Perimeter</b>	A 30 nm x 30 nm area surrounding the target area modeled in accordance to the following criteria: (a) Elevation grid post spacing, 10 m. (b) Terrain imagery resolution, 0.5 m. (c) Contains all objects of height greater than, 15 m (50 ft).
<b>Area, Target</b>	A 6 nm x 6 nm area surrounding the target (see “Target” glossary entry) modeled in accordance to the following criteria: (a) Elevation grid post spacing, 10 m. (b) Terrain imagery resolution, 0.5 m. (c) Contains all objects of height greater than, 3.3 m (10 ft).
<b>Articulated Part</b>	A child part of a model (usually a moving model) that is allowed some degree of motion with respect to the parent body of the model. Examples of articulated parts include tank turrets, refueling drogues, and helicopter rotor blades.
<b>B</b>	
<b>Background Area</b>	See Area, Background.
<b>Base Material</b>	See Material, Base.
<b>Base Material Table (BMT)</b>	A data structure that contains a description of the Base Materials available to the CDB. There is only one BMT per CDB. Each entry in the BMT corresponds to a Base Material; the entry associates a name to the Base Material.
<b>C</b>	
<b>CDB-compliant device</b>	A device that can directly input synthetic environment data that conforms to the format, structure and conventions of this Specification. A device need not input and process all of the CDB datasets and attributes to be CDB-compliant.
<b>CDB-compliant simulator</b>	A simulator whose client-devices can input synthetic environment data that conforms to the format, structure and conventions of this Specification. Any subsystem or client-device that does not natively conform to this Specification requires that CDB data be formatted and structured to the device’s native format and structure through the use of a runtime publisher. A simulator need not input and process all of the CDB datasets and attributes to be CDB-compliant.



	Description
<b>CDB Translator</b>	An off-line software process that translates an environmental database from its toolset-native format(s) to the CDB format.  <b>NOTE:</b> The CDB data formats are based on industry standard tool formats; as a result a translation to the formats prescribed by this Specification may not be required.
<b>CDB Server</b>	Gateway platform to CDB mass storage and applicable infrastructure. The CDB servers access, filter and distribute CDB data in response to requests from the Database Generation Facility (DBGF) and the client-devices (or their runtime publishers).
<b>CDB Geocell</b>	Earth area, aligned to lines of latitude and longitude in accordance with Table 2-2: Size of CDB Geocell per Zone.
<b>Channel</b>	A field of view segment within a visual system's total viewing field for which a corresponding unique scene segment is calculated and presented.
<b>Client application</b>	A software application that requires a complete or partial synthetic representation of the world. CDB applications may require a CDB runtime publisher to convert the CDB DB into a form it can directly input. Used interchangeably with "Client-device" in this specification.
<b>Client-device</b>	Simulation sub-systems (Image Generators (IGs), Radar, Weather Server, Computer Generated Forces (CGF) Terrain Server, etc.) that require a complete or partial synthetic representation of the world. A CDB client-device may require a CDB runtime publisher to convert the CDB data into a form it can ingest.
<b>Coordinate System</b>	A system of notation, usually Three-Dimensional (3D), by which the position of any point can be defined. Can be spherical, geodetic, ellipsoid or Cartesian. Must have a point of origin (where all axes have a magnitude of zero), an orientation, and a convention for translation along the axes.
<b>Composite Material</b>	See Material, Composite.
<b>Composite Material Table</b>	A data structure that contains a description of the Composite Materials available in a CDB tile or on a model. There is only one CMT per CDB tile or model. Each entry in the CMT corresponds to a Composite Material.



	<b>Description</b>
<b>Coordinate System, Geocentric</b>	Coordinates used to define points in a Geocentric Spatial Frame (see Spatial Frame, Geocentric).
<b>Coordinate System, Geographic</b>	The Geographical coordinate system is the most commonly used coordinate system to represent earth surfaces. The coordinates are longitude, latitude and altitude above mean sea level. The reference data is based on the WGS-84 ellipsoid, i.e., geographical latitude $\phi$ and longitude $\lambda$ are the angles of the normal on the WGS-84 reference ellipsoid along the point to the equator and zero meridian. The angles are given as degrees, minutes and seconds. Altitude above mean sea level is the distance above and normal to the ellipsoid in meters. The WGS-84 ellipsoidal earth models provides for accurate calculations over long distances on the earth's surface. The WGS-84 earth model represents a good approximation of the shape of the earth over the smoothed mean sea level (geoid gravitational equipotential) to within about one hundred meters.
<b>Correlation, Algorithmic</b>	The degree of informational consistency between the outputs of two or more devices with equivalent Arithmetic Logic Units, each submitted to the same input data. (e.g., consider two devices meshing terrain from a regular grid of elevation points, one using a regular mesh of right-handed triangles using the elevation points as vertices, and the other with a DeLauney triangulated mesh derived from the grid of elevation points).
<b>Correlation, Numerical</b>	The degree of informational consistency between the outputs of two or more devices, each submitted to the same input data, (e.g., two devices computing the sine of an angle, one with a series of 10 terms, and another with an interpolation of a look-up table with 100 entries, or one device using 32-bit signed integers for its internal computations and the other using single-precision floats). The CDB Specification addresses Runtime Source Database numerical accuracy correlation errors because a single representation is used for each data set.
<b>Correlation, Parametric</b>	The degree of informational consistency between the outputs of two or more devices, each submitted to the same input data but to different control parameters (e.g., consider two devices generating regular meshes of right-handed triangles based on a regular grids of elevation points organized by Level-of-Detail (LOD), one using an LOD meshing tolerance parameter of 1m and the other 2m).

	Description
<b>Correlation, Raw Source DB</b>	The degree of informational consistency between two or more sets of raw data <sup>124</sup> (i.e., inputs to a modeling station) representing aspects of the same environment (for instance, the correlation errors arising from Digital Terrain Elevation Data (DTED) elevation data that does not perfectly match the satellite raster imagery due to oblique view distortions induced by the satellite). Correlation errors are intrinsic to the process of gathering data because there is no means to gather <u>all</u> of the required data from a single device, at a single instant in time. Instead, datasets (e.g., elevation, raster imagery, geometry) are each gathered from various devices of various types, at different times, etc. As a result, this process inherently introduces (raw source) correlation errors.
<b>Correlation, Runtime DB</b>	The degree of informational consistency between two or more runtime databases representing the same synthetic environment. The CDB Specification eliminates database correlation errors since only one database is used to represent the same synthetic environment. A runtime database is a device-loadable database format that can be processed by a target device. The CDB Specification defines a format that can be entered in runtime by client-devices that conform to the Specification. By definition, the CDB Specification addresses all runtime database-level correlation error.
<b>Correlation, Source DB</b>	The degree of informational consistency between the internal datasets of a source database produced by a DB generation toolset. To a large extent, the effort expended by a modeler at their DB workstation consists in eliminating (or at least reducing) correlation errors arising from miss-correlated raw source data.
<b>Corridor Area</b>	See Area, Corridor.
<b>Culture</b>	See Feature, Cultural.
<b>Culture, 2D</b>	Short for 2D Cultural Feature. The 2D representation of a man-made or natural object (such as a road, a runway, a forest canopy), conformed to the terrain.

<sup>124</sup> In this context, raw source denotes any input to the modeling workstation that is used to assemble the synthetic environment; consequently, the data may have undergone some level of post-processing (such as image color-balancing, image ortho-rectification, etc.) or may be in a specific source interchange format (such as SIF, SEDRIS, etc.)



	<b>Description</b>
<b>Culture, 3D</b>	Short for 3D Cultural Feature. The 3D representation of erect man-made or natural object (such as buildings, trees, towers) positioned on top of and usually conformed to the terrain.
<b>D</b>	
<b>Data Dictionary</b>	In database management systems, a file that defines the basic organization of a database. Data dictionaries explicitly define data content and how to access this content within a binary file (for example, a data dictionary may contain a list of all files in the database, the number of records in each file, and the names and types of each field). Most database management systems keep the data dictionary hidden from users to prevent them from accidentally destroying its contents. Data dictionaries do not contain any actual data from the database, only book keeping information for managing it. Since there are no widely accepted standards for data dictionaries, most of the CDB data content and structure are explicitly defined by this CDB Specification rather than by the metadata held within a Data Dictionary file.
<b>Data Duplication</b>	Data logically representing the same information, copied one or more times within a complex data structure. The CDB Specification eliminates all duplication of data, i.e., the data appears once and only once within the CDB structure.
<b>Data Normalization</b>	Data that has been manipulated to eliminate informational redundancy and/or data duplication and anomalies. The normalization process ensures internal consistency, minimizes informational redundancy, and maximizes stability (associates attributes with entities based on the inherent properties of the data rather than on the application requirements).
<b>Data Redundancy</b>	Information in the form of data that can readily be derived (within the limits of technological/cost reasonability) and re-formatted, or re-derived from other data.
<b>Dataset</b>	Organized group of related environmental data that cannot be broken down into a smaller set used to describe a synthetic environment element of the world.

	Description
<b>Database Fidelity</b>	Reflects the amount and type of synthetic environmental data needed by client-devices to simulate real-world environmental data with greater fidelity. Consider for instance a simulator client-device capable of supporting a single-surface earth skin representation versus one capable of representing a multi-surface earth skin that represent tunnels, bathymetric data, location-dependent tide heights, etc.
<b>Database Assembly</b>	In many database tools, the generation of the terrain plays a pivotal role in the database assembly process because all of the cultural features are conformed and constrained to the terrain representation and structure. Most client-devices in existence today have interdependent terrain geometry, raster imagery and culture; as a result of this, most tools in use today resolve these inter-dependencies during this critical and computationally expensive database assembly step.
<b>Database Generation Facility (DBGF)</b>	A geo-graphically co-located group of workstation(s), computer platforms, input devices (digitizing tablets, etc.), output devices (stereo viewers, etc.), modeling software, visualization software, CDB Server, CDB off-line publishing software and any other associated software and hardware used for the development/modification of the CDB. The DBGF is used for the purpose of CDB creation and CDB updates. Each workstation is equipped with one or more specialized tools. The tool suite provides the means to generate and manipulate the synthetic environment.

	Description
<b>Database Generation Timeline</b>	Elapsed time from availability of Environmental DB requirements (geographic extent, fidelity, resolution, etc.) to availability of a compliant runtime Environmental DB, ready for use on all client-devices of a simulator.
<b>Database Publishing</b>	<p>A process (either off-line<sup>125</sup> or on-line) where all of differences between the tool-native representation and the client-device internal representation of the synthetic environment database are resolved. During this step, the publisher transforms the assembled database so that it satisfies the client-device's:</p> <ul style="list-style-type: none"> <li>▪ internal formats</li> <li>▪ internal data structure and organization</li> <li>▪ internal naming conventions</li> <li>▪ internal precision and number representation</li> <li>▪ data fidelity requirements (typically parameters that match the client-device algorithms)</li> <li>▪ performance limitations</li> <li>▪ level-of-detail representation and conventions</li> </ul>
<b>Database Publishing, Offline</b>	The process of performing the steps listed above in Database Publishing, and then storing the result in a distinct SE database for each client-device. (Note that the stored databases are different for each client-device type and each vendor type). In many cases, the published databases are proprietary.
<b>Database Publishing, Online</b>	The process of performing the steps listed above in Database Publishing, on-the-fly, based on paging requests of a client-device. Since the publishing is performed on-demand, it exists only momentarily in memory; it is not stored on disk.
<b>Database Resolution</b>	Informational density (for instance, the number of elevation values per km <sup>2</sup> , pixels per km <sup>2</sup> , polygons per km <sup>2</sup> ) of a modeled dataset.
<b>Data Precision</b>	Corresponds to the numerical precision (i.e., number of bits allocated) used to represent a unit.
<b>Directional Illumination</b>	See Illumination, Directional.
<b>E</b>	

<sup>125</sup> When applied as an off-line process, the term “compilation” is often used instead.

	Description
<b>Environmental Data Coding Specification</b>	An Environmental Data Coding Specification provides a mechanism to specify the environmental "things" that a particular data model is intended to represent. That is, a feature such as building could be represented alternatively as a Man-made Point Feature, a Shapefile Radar RCS polar diagram or as an OpenFlight model, or some combination of these. The representation of these is chosen by the data modeler and is orthogonal to the semantic of the "thing" that is represented (and its location). The provision of such a "thing" results in a shared understanding of "what the thing is and what it potentially means" to all participating applications.
<b>Environmental Data Representation Model</b>	A data representation model (EDRM) is a description used to provide identification of all environmental data elements within a system, including their attributes and the logical relationships between data elements.
<b>Environmental DB, Runtime</b>	A device-loadable database format that can be processed by a target device. The CDB Specification defines a format that can be entered in runtime by simulator client-devices that conform to the Specification.
<b>Eyepoint</b>	A single point (monocular) location of the observer's eye relative to a scene representation. Usually a point within the cockpit of the simulated aircraft or vehicle.
<b>Eyepoint, Pilot</b>	The normal eyepoint position when the pilot's seat is adjusted properly for flying the aircraft. Defined by the aircraft manufacturer for each pilot seating position. The eyepoint(s) for which the display design is normally optimized and typically used for display testing purposes.
<b>F</b>	
<b>Feature Set</b>	Describes a set of logically related datasets. Example: a Terrain Feature set is comprised of the Elevation, Imagery, MinElevation, MaxElevation, etc. datasets.
<b>Feature, Areal</b>	A representation of closed area-oriented features conformed relative to the terrain such as forested areas, fields. The information includes areal feature type identification, location, orientation, 2D geometry, connectivity, attribution and other surface characteristics relevant to simulation.
<b>Feature, Cultural</b>	A generalization of point, lineal and areal features.





	Description
<b>Feature, Linear</b>	The representation of predominantly man-made multi-segmented line-oriented features conformed relative to the terrain (such as runways, roads, transmission lines, fences). The information includes linear feature type identification, location, orientation, lineal geometry, connectivity, attribution and other surface characteristics relevant to simulation.
<b>Feature, Point</b>	The representation of a single location in space or on the earth's surface. It consists of a single <latitude, longitude> coordinate with or without an elevation. When a point feature does not have an elevation, it is deemed to be on the surface of the earth. It is often associated with a 3D model. The information includes point-feature type identification, location, orientation, connectivity, attribution and other characteristics relevant to simulation.
<b>Flat Earth</b>	Jargon used in simulation community to signify the projection of the earth ellipsoid onto a flat surface. The flat Earth approximation retains terrain relief but eliminates the effects of Earth surface curvature. If you stay in the vicinity of a given fixed point, it may be a good enough approximation to consider the earth as "flat", and use a North, East, Down rectangular coordinate system with origin at the fixed point.
<b>FOV, Field of View</b>	The horizontal and vertical subtended angles from a designated eyepoint to the boundaries of a visual system channel (channel FOV) or all channels (system FOV).
<b>G</b>	
<b>Geocell</b>	Short form for geographic cell. A 1° of latitude by 1° of longitude area on the surface of the earth. At the equator, this corresponds to an area of approximately of 111,319m × 111,319m. (See also CDB Geocell).
<b>Geocentric Spatial Frame</b>	See Spatial Frame, Geocentric.
<b>Geographic Extent</b>	An earth surface area that has been modeled.
<b>Geographic Projection</b>	Geographic projections are a way of showing the curved surface of the Earth on a flat surface like a piece of paper.
<b>Geospecific Model</b>	A model is said to be geospecific if it is instanced once and only once within a CDB. Geospecific models usually correspond to unique (in either shape, size, texture, materials or attribution), man-made, real-world 3D cultural features.

	Description
<b>Geotypical Model</b>	A model is said to be geotypical if it instanced multiple times within a CDB. Geotypical models correspond to representative (in shape, size, texture, materials and attribution) models of real-world man-made or natural 3D cultural features.
<b>H</b>	
<b>HLZ Area</b>	See Area, HLZ.
<b>I</b>	
<b>Illumination</b>	One or more sources of illumination for the objects in the scene, such as daylight, twilight, landing lights.
<b>Illumination, ambient</b>	The non-directional component of illumination for the scene. Daylight, twilight and moonlight have ambient components over the entire scene. Landing lights typically provide ambient-type illumination over a limited area of the scene.
<b>Illumination, Directional</b>	Scene illumination provided by an illumination source at a particular position or direction in the environmental database spatial frame. The effect on object luminance depends on the angle between the illumination source and object surface normal.
<b>J</b>	
<b>K</b>	
<b>L</b>	
<b>Latency</b>	The time interval from a request to a prescribed response from the targeted device.
<b>Level-of-Detail (LOD)</b>	Representations of the same thing that differ only in the amount of fidelity. An LOD is said to be coarse if it contains little detail or fine if it contains considerable detail.
<b>Light Point</b>	A database element used to model a point source of light (e.g., a taxiway lights, street lights, collision lights).
<b>Light String</b>	A group of lights, usually a series of light points sharing common spacing characteristics and are of a common type.
<b>Lineal Feature</b>	See Feature, Lineal.



	Description
<b>Local Vertical Spatial Frame</b>	See Spatial Frame, Local Vertical.
<b>M</b>	
<b>Material</b>	Shorthand for either Base Material or Composite Material.
<b>Material, Base</b>	Symbolic representation of a basic material in the CDB. Basic materials are inputs to production or manufacturing processes. They are often raw, that is unprocessed, but are sometimes processed before being used in more advanced production processes. Basic materials represent the substances out of which a thing is or can be made. Examples are materials such as steel, aluminum, copper, sand, soil, stone, glass, concrete, wood, water, rubber. Base materials are chosen for their relevance to simulation.
<b>Material, Composite</b>	A symbolic representation that corresponds to a composite material that is made up of a primary substrate and one or more secondary substrates. Each substrate is composed of one or more base materials entries. The substrates can each be assigned a thickness.
<b>Metadata</b>	Data about data. Metadata describes how and when and by whom a particular set of data was collected, and how the data is formatted.
<b>Mission Functions</b>	A set of low-level simulator query functions performed on the synthetic environment; includes such functions as Height Above Terrain (HAT), Height Above Culture (HAC), Collision Detection (CD), Line-Of-Sight (LOS), Laser Range Finder (LRF), etc.
<b>Model</b>	A term which stands for the 2D or 3D representation of features (exclusive of the terrain and/or bathymetry itself) within the synthetic environment database. Models can be statically positioned on the terrain (i.e., a cultural feature), or they are freely moving (i.e., a moving model). Models are often a 3D representation of a man-made or a natural object positioned and conformed relative to the terrain. The information includes its geometry, articulations, raster imagery (texture, normal map, light map, etc.), lighting systems, and other characteristics relevant to simulation.
<b>Model, 2D</b>	Refers to the modeled representations of 2D features; i.e., lineal or areal features that have no significant height with respect to the underlying terrain; 2D Models generally conform to the terrain profile.

	Description
<b>Model, 3D</b>	Refers to the modeled representation of 3D features that can be readily distinguished from the underlying terrain. In the case where they are unique, they are referred to as GSModels. In the case where they are instanced, they are referred to as GTModels. 3DModels capable of movement are called MModels. In the case where MModels are positioned by the modeler, they are called statically-positioned MModels.
<b>Model, Cultural</b>	A model that is statically positioned on the terrain or bathymetry skin. Cultural models are often a 3D representation of a man-made or a natural object positioned and conformed relative to the terrain. The information includes its geometry, articulations, raster imagery (texture, normal map, light map, etc.), lighting systems, and other characteristics relevant to simulation.
<b>Model, GS</b>	The geospecific representation of a cultural feature that is unique within the CDB.
<b>Model, GT</b>	The geotypical representation of a cultural feature that can be reused several times throughout the CDB.
<b>Model, Moving</b>	A model that is not fixed at one location in the synthetic environment database. The simulation host can update the position and orientation of a moving model at every simulation iteration cycle. A moving model is a 3D representation of man-made and natural objects free to move within the CDB. The information includes feature type identification, (vehicle class, type, model, etc.), geometry, articulations, raster imagery (texture, normal map, light map, etc.), lighting systems, connectivity to special effects, attribution and other characteristics relevant to simulation.
<b>Model-LOD</b>	Refers to a specific level of detail of the modeled representation of a feature; a general term encompassing both 2D and 3D Model-LOD
<b>Model-LOD, 2D</b>	Refers to a specific level of detail of a 2D model.
<b>Model-LOD, 3D</b>	Refers to a specific level of detail of a 3D model.
<b>Modeler</b>	The person who creates and assembles a synthetic environment database.
<b>N</b>	



	Description
<b>Navigational Data</b>	Is a representation of ARINC-424 and DAFIF data in the form of NAVAIDs (VHF, ILS/MLS, NDB, Markers), Communications Stations, Airport/Heliport (including SIDs, STARs, Terminal Procedure/Approaches, Gates), Runway/Helipad, Waypoints, Routes, Holding Patterns, Airways and Airspace.
<b>Normal Vector</b>	A vector of unit length perpendicular to a surface.
<b>Numerical Correlation</b>	See Correlation, Numerical.
<b>O</b>	
<b>Ownship</b>	The vehicle (aircraft, ship, tank, etc.) being simulated.
<b>P</b>	
<b>Parametric Correlation</b>	See Correlation, Parametric.
<b>Pilot Eyepoint</b>	See Eyepoint, Pilot.
<b>Point Feature</b>	See Feature, Point.
<b>Q</b>	
<b>R</b>	
<b>Raw Source</b>	A term generally used to describe the data imported into the database generation workstation for the purpose of off-line assembling and building the synthetic environment. The level of pre-processing applied to the source may vary considerably (from raw data directly from sensors such as unprocessed satellite raster imagery or photos to data directly usable by simulator client-devices). Source data need not be in digital form (e.g., photos).
<b>Raw Source DB Correlation</b>	See Correlation, Raw Source DB.
<b>Runtime Publisher</b>	A real-time software process (either shared or dedicated to a computer platform) that a simulation application client-device uses to transform or translate CDB data into a format that can be directly input by the client-device it serves.

	Description
<b>Runtime DB Correlation</b>	See Correlation, Runtime DB.
<b>S</b>	
<b>Sensor Environmental Model (SEM)</b>	A simulation of the synthetic environment over a portion of the electromagnetic spectrum that is relevant to a client-device. A SEM is usually based on mathematical model of the environment for the portion of the electromagnetic spectrum of interest.
<b>Sensor Simulation Model (SSM)</b>	A simulation of a real-world sensor over a portion of the electromagnetic spectrum that is relevant to the sensor being simulated. A SSM is usually based on mathematical model of the real-world sensor for the portion of the electromagnetic spectrum of interest.
<b>Simulator CDB Repository</b>	The simulator CDB repository consists of a mass storage system (typically a storage array) and its associated network infrastructure. It is connected to the UMC (primarily for update purposes) and the CDB Servers (for simulator client-device runtime access).
<b>Source DB Correlation</b>	See Correlation, Source DB.
<b>Spatial Coordinate System</b>	A spatial coordinate system is a means of associating a unique coordinate with a point in object-space.
<b>Spatial Frame (SF)</b>	A spatial reference frame is a spatial coordinate system for a region of object-space.
<b>Spatial Frame, Geocentric</b>	An earth-centered spatial frame that defines 3D Euclidian space with respect to the geometric centre of the reference ellipsoid, the centre of the earth <sup>126</sup> . The reference datum of the earth-centered SF is based on the WGS-84 ellipsoid reference model. In this SF, the z-axis is pointing at the North Pole, the x-axis is pointing at the intersection of the equator and the Greenwich meridian, the prime meridian, and the y-axis is pointing at the intersection of the equator and 90 degrees east longitude.

<sup>126</sup> In this coordinate system, the z-axis is defined by the earth's axis of rotation. Furthermore, the equatorial plane is defined by a plane normal to the z-axis, halfway from the North and South Pole (using gravitational equipotential).



	Description
<b>Spatial Frame, Local Vertical</b>	The Local Vertical Coordinate System (LVCS) SF defines a 3D Euclidian space. It is SF similar to the Geocentric SF except that the origin of the SF is translated and rotated to a point on the surface of the WGS-84 ellipsoid. At that point, the x-y plane is tangent to the surface of the earth and the z-axis is normal to the ellipsoid.
<b>T</b>	
<b>Target</b>	A 1 nm x 1 nm area modeled in accordance to the following criteria: (a) Elevation grid post spacing, 3 m. (b) Terrain imagery resolution, 0.25 m. (c) Contains all objects of height greater than, 1 m (3 ft).
<b>Target Area</b>	See Area, Target.
<b>Target Perimeter Area</b>	See Area, Target Perimeter.
<b>Terrain</b>	A representation of earth surface shape/elevation, raster imagery, surface attribution and other earth surface characteristics relevant to simulation. Also includes bodies of water such as oceans, lakes.
<b>Terrain Profile</b>	See Terrain.
<b>Terrain Skin</b>	See Terrain.
<b>U</b>	
<b>V</b>	
<b>Viewpoint</b>	The viewpoint is the position from which the synthetic environment database is being observed.
<b>W</b>	
<b>X</b>	
<b>Y</b>	
<b>Z</b>	
<b>0-9</b>	
<b>2D Feature</b>	See Culture, 2D.



	Description
<b>2D Model</b>	See Model, 2D
<b>2D Model-LOD</b>	See Model-LOD, 2D
<b>3D Feature</b>	See Culture, 3D.
<b>3D Model</b>	See Model, 3D
<b>3D Model-LOD</b>	See Model-LOD, 3D



## Chapter 9

### 9 Acronyms and Abbreviations

	Definition
<b>A</b>	
<b>ADF</b>	Automatic Direction Finder
<b>ATS</b>	Air Traffic Simulation
<b>B</b>	
<b>BMT</b>	Base Material Table
<b>BW</b>	Black and White
<b>C</b>	
<b>CD</b>	Collision Detection
<b>CDT</b>	Complex Data Type
<b>CDB</b>	Common Database (for Real-time Simulation)
<b>CGF</b>	Computer Generated Forces
<b>COMMS</b>	Communication Systems
<b>COTS</b>	Commercial-off-the-shelf
<b>CMT</b>	Composite Material Table
<b>CPU</b>	Central Processing Unit
<b>D</b>	
<b>DAFIF</b>	Digital Aeronautical Flight Information File
<b>DB</b>	Database
<b>DBGF</b>	DataBase Generation Facility
<b>DFAD</b>	Digital Feature Analysis Data
<b>DGIWG</b>	Digital Geographic Information Working Group
<b>DIGEST</b>	Digital Geographic Exchange Standard
<b>DRM</b>	Data Representation Model
<b>DTED</b>	Digital Terrain Elevation Data
<b>DNS</b>	Doppler Navigation System
<b>E</b>	



	<b>Definition</b>
<b>EDCS</b>	Environmental Data Coding Specification
<b>EDRM</b>	Environment Data Representation Model
<b>EW</b>	Electronic Warfare
<b>ERM</b>	Earth Reference Model
<b>F</b>	
<b>FACC</b>	Features and Attributes Catalog Codes
<b>FC</b>	Fiber Channel
<b>FDD</b>	Feature Data Dictionary
<b>FFS</b>	Full Flight Simulator
<b>FLIR</b>	Forward Looking InfraRed
<b>FMS</b>	Full Mission Simulator
<b>FSC</b>	FACC Sub Code
<b>FsE</b>	Fast Ethernet
<b>ft</b>	Feet
<b>G</b>	
<b>GPS</b>	Global Positioning System
<b>GB</b>	GigaByte
<b>GbE</b>	Gigabit Ethernet
<b>H</b>	
<b>HAC</b>	Height Above Culture
<b>HAO</b>	Height Above Ocean
<b>HAT</b>	Height Above Terrain
<b>HLA</b>	High Level Architecture
<b>HLZ</b>	Helicopter Landing Zone
<b>HOC</b>	Height Of Culture
<b>HOO</b>	Height Of Ocean
<b>HOT</b>	Height Of Terrain
<b>Hz</b>	Hertz
<b>I</b>	

	Definition
<b>ICAO</b>	International Civil Aviation Organization
<b>ICD</b>	Interface Control Document
<b>ID</b>	Identification
<b>IEEE</b>	Institute of Electrical and Electronics Engineers
<b>IG</b>	Image Generator
<b>ILS</b>	Instrument Landing System
<b>INU</b>	Inertial Navigation Unit
<b>IO</b>	Input Output
<b>IP</b>	Internet Protocol
<b>IP</b>	Intellectual Property
<b>IR</b>	InfraRed
<b>J</b>	
<b>JPEG</b>	Joint Photographic Expert Group
<b>K</b>	
<b>Km</b>	kilometers
<b>L</b>	
<b>LOD</b>	Level of Detail
<b>LOS</b>	Line of Sight
<b>LRF</b>	Laser Ranging Function
<b>LVCS</b>	Local Vertical Coordinate System
<b>LZW</b>	Lempel-Ziv-Welch (compression algorithm)
<b>M</b>	
<b>m</b>	meter(s)
<b>MB</b>	MegaByte
<b>MIF</b>	Mission Functions
<b>MIP</b>	Multium In Parvo (“many in a small place”)
<b>MLS</b>	Microwave Landing System
<b>MR</b>	Mission Rehearsal
<b>N</b>	



	<b>Definition</b>
<b>NAV</b>	Navigation System
<b>NAVSTAR</b>	Navigation System Using Timing and Ranging
<b>NDB</b>	Non-Directional Beacons
<b>NIMA</b>	National Imagery and Mapping Agency
<b>nm</b>	nanometer
<b>NVG</b>	Night Vision Goggle
<b>O</b>	
<b>OneSAF</b>	One Semi-Automated Forces
<b>OS</b>	Operating System
<b>OTW</b>	Out The Window
<b>P</b>	
<b>PB</b>	PetaByte
<b>PLS</b>	Personnel Locating System
<b>Q</b>	
<b>R</b>	
<b>RCS</b>	Radar Cross Section
<b>RGB</b>	Red Green Blue
<b>RF</b>	Radio Frequency
<b>ROI</b>	Region of Interest
<b>RTP</b>	Run Time Publisher
<b>S</b>	
<b>SAF</b>	Semi-Automated Forces
<b>SE</b>	Synthetic Environment
<b>SEDRIS</b>	Synthetic Environment Data Representation & Interchange Specification
<b>SEM</b>	Sensor Environment Model
<b>SID</b>	Standard Instrument Departure
<b>SIF</b>	Standard Interchange Format
<b>SF</b>	Spatial Frame
<b>SM</b>	Spatial Model

	Definition
<b>SSM</b>	Sensor Simulation Model
<b>STAR</b>	Standard Terminal Arrival Route
<b>T</b>	
<b>TACAN</b>	Tactical Air Navigation
<b>TB</b>	TeraByte
<b>U</b>	
<b>UHRB</b>	Ultra High Resolution Buildings (OneSAF)
<b>UM</b>	Update Manager
<b>UMS</b>	Update Manager Server (software)
<b>UMC</b>	Update Manager Client (software)
<b>UML</b>	Unified Modeling Language
<b>UTM</b>	Universal Transverse Mercator
<b>V</b>	
<b>VHF</b>	Very High Frequency
<b>VOR</b>	VHF Omni Range
<b>VSTI</b>	Visible Spectrum Terrain Imagery
<b>VSTLM</b>	Visible Spectrum Terrain Light Map
<b>W</b>	
<b>WGS</b>	World Geodetic System
<b>WX</b>	Weather or Weather Simulation
<b>X</b>	
<b>XML</b>	Extensible Mark-up Language
<b>Y</b>	
<b>Z</b>	
<b>0-9</b>	
<b>1D</b>	Unidimensional
<b>2D</b>	Bidimensional
<b>3D</b>	Three-Dimensional





## Chapter 10

### 10 Reference Documents

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: <a href="http://www.digest.org/">http://www.digest.org/</a>
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 <i>Simulation Interoperability Standards Organization</i> at <a href="http://www.sisostds.org/">http://www.sisostds.org/</a>
5	Extensible Markup Language (XML) 1.0 (Third Edition)	Bray, Tim, et al. <a href="http://www.w3.org/TR/2004/REC-xml-20040204/">http://www.w3.org/TR/2004/REC-xml-20040204/</a> W3C Recommendation, February 04, 2004.
6	Guide - PD6777, BSI's <i>Guide to the practical implementation of JPEG 2000</i>	The document can be found at: <a href="http://www.jpeg.org/">http://www.jpeg.org/</a> Other useful sites include: <a href="http://en.wikipedia.org/wiki/SRGB_color_space">http://en.wikipedia.org/wiki/SRGB_color_space</a> 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



Ref	Title	Description
9	MIL-STD-2411 Raster Product Format Specification	<p>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.</p> <p>Department of Defense Information Technology Standards Registry Baseline Release 04-2.0.</p> <p><a href="http://earth-info.nga.mil/publications/specs/printed/2411/2411_RPF.pdf">http://earth-info.nga.mil/publications/specs/printed/2411/2411_RPF.pdf</a></p>
10	MIL-C-89041 Controlled Image Base Specification	<p>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.</p> <p><a href="http://www.fas.org/irp/program/core/mil-c-89041-cib.htm">http://www.fas.org/irp/program/core/mil-c-89041-cib.htm</a></p>
11	OpenFlight Scene Description Database Standard, Version 16.0, Revision A, November 2004, Presagis Inc	<p>The original document has been annotated by CAE to create the CDB-annotated OpenFlight Standard.</p>
12	Product Standard for the Digital Aeronautical Flight Information File (DAFIF), Eight Edition, Doc. # PS/1FD/086	<p>National Imagery and Mapping Agency (NIMA), April 2003.</p>
13	SEDRIST™ - Synthetic Environment Data Representation Interchange Specification	<p>The Source for Synthetic environment Representation and Interchange.</p> <p><a href="http://www.sedris.org">http://www.sedris.org</a></p>
14	Shapefile Technical Description - an ESRI White Paper—July 1998	<p>The original document has been annotated by CAE Inc to create the CDB-annotated Shapefile Technical Description.</p>
15	<b>The SGI Image File Format</b> , Version 1.00, Paul Haeberli, Silicon Graphics Computer Systems	<p>This specification can be found at:</p> <p><a href="http://paulbourke.net/dataformats/sgirgb/sgiversion.html">http://paulbourke.net/dataformats/sgirgb/sgiversion.html</a></p>
16	TIFF rev 6.0 Adobe Developers Association, Adobe Systems Incorporated, 1585 Charleston Road and P.O. Box 7900 Mountain View, CA 94039-7900	<p>A copy of this original standard can be found at:</p> <p><a href="http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf">http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf</a></p> <p>and at:</p> <p><a href="ftp://ftp.adobe.com/pub/adobe/DeveloperSupport/TechNotes/PDFfiles">ftp://ftp.adobe.com/pub/adobe/DeveloperSupport/TechNotes/PDFfiles</a></p> <p>The original document has been annotated by CAE Inc to create the CDB-annotated TIFF Standard.</p>

Ref	Title	Description
17	XML Schema Part 0: Primer Second Edition	Fallside, David, Priscilla Walmsley. <a href="http://www.w3.org/TR/xmlschema-0/">http://www.w3.org/TR/xmlschema-0/</a> W3C Recommendation, October 28, 2004.
18	XML Schema Part 1: Structures Second Edition	Thompson, Henry S., et al. <a href="http://www.w3.org/TR/xmlschema-1/">http://www.w3.org/TR/xmlschema-1/</a> W3C Recommendation, October 28, 2004.
19	XML Schema Part 2: Datatypes Second Edition	Biron, Paul V., Ashok Malhotra. <a href="http://www.w3.org/TR/xmlschema-2/">http://www.w3.org/TR/xmlschema-2/</a> W3C Recommendation, October 28, 2004.
20	ICAO Airline Designator	List of ICAO Airline Codes, <a href="http://en.wikipedia.org/wiki/Airline_codes">http://en.wikipedia.org/wiki/Airline_codes</a>
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:  <a href="http://aircraftdesign.nuua.edu.cn/lo/Ref/General%20Topics/radar_signatures_and_relations_to_rcs.pdf">http://aircraftdesign.nuua.edu.cn/lo/Ref/General%20Topics/radar_signatures_and_relations_to_rcs.pdf</a>
22	AN/APA to AN/APD - Equipment Listing.	This document can be obtained at the following Internet address:  <a href="http://www.designation-systems.net/usmilav/jetds/an-apa2apd.html#_APA">http://www.designation-systems.net/usmilav/jetds/an-apa2apd.html#_APA</a>
23	Radar Polarimetry - Fundamentals of Remote Sensing. National Resources Canada.	This document can be obtained at the following Internet address:  <a href="https://www.nrcan.gc.ca/earth-sciences/geomatics/satellite-imagery-air-photos/satellite-imagery-products/educational-resources/9275">https://www.nrcan.gc.ca/earth-sciences/geomatics/satellite-imagery-air-photos/satellite-imagery-products/educational-resources/9275</a>
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:  <a href="http://www.mar-it.de/Radar/RCS/RCS_xx.pdf">http://www.mar-it.de/Radar/RCS/RCS_xx.pdf</a>
25	Decibels relative to a square meter – dBsm. By Zhao Shengyun.	This document can be obtained at the following Internet address:  <a href="http://radarproblems.com/chapters/ch06.dir/ch06pr.dir/c06p11.dir/c06p11.htm">http://radarproblems.com/chapters/ch06.dir/ch06pr.dir/c06p11.dir/c06p11.htm</a>
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



Ref	Title	Description
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 <i>Simulation Interoperability Standards Organization</i> at the following address: <a href="http://www.sisostds.org/">http://www.sisostds.org/</a>
32	MIL-PRF-89039 Amendment 2	Performance Specification Vector Smart Map (VMAP Level 0), 28 September 1999 <a href="http://en.wikipedia.org/wiki/Vector_Map">http://en.wikipedia.org/wiki/Vector_Map</a> <a href="http://earth-info.nga.mil/publications/specs/printed/VMAP0/vmap0.html">http://earth-info.nga.mil/publications/specs/printed/VMAP0/vmap0.html</a>
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), 1 <sup>st</sup> August, 2002
35	OneSAF Ultra High Resolution Building (UHRB) Object Model	OneSAF UHRB Object Model (Version 2.2, Document Revision E, March 7 <sup>th</sup> , 2008, Contract #: N61339-00-D-0710, Task Order: 28.) <a href="http://www.onesaf.net/community/systemdocuments/v.3.0/MaintenanceManual/erc/UHRB_2_Object_Model.pdf">http://www.onesaf.net/community/systemdocuments/v.3.0/MaintenanceManual/erc/UHRB_2_Object_Model.pdf</a>
36	OneSAF Ultra High Resolution Building (UHRB) On-Disk Format	OneSAF UHRB On-Disk Format Model (Version 2.2, Document Revision E, March 7 <sup>th</sup> , 2008, Contract #: N61339-00-D-0710, Task Order: 28.) <a href="http://www.onesaf.net/community/systemdocuments/v.3.0/MaintenanceManual/erc/UHRB_2_On_Disk_Format.pdf">http://www.onesaf.net/community/systemdocuments/v.3.0/MaintenanceManual/erc/UHRB_2_On_Disk_Format.pdf</a>
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 14 <sup>th</sup> February, 2008

## Chapter 11

### 11 List of Contributors

The CDB Specification has been developed from active collaboration between many individuals from different organization levels. They are listed below in alphabetical order.

#### **CDB Authors**

Bernard Lalonde, CAE  
Bernard Leclerc, CAE  
Michel Lagacé, CAE  
Pierre Samson, CAE

#### **CDB Contributors and Reviewers**

Andrew Fernie, CAE  
Arnaud Banel, CAE  
Brian Ford, FSI  
David Nadeau, Presagis  
Frédéric St-Laurent, Presagis  
Hermann Brassard, Presagis  
Jay Freeman, CAE  
John Hortenstine, FSI  
John Oliver, Presagis  
Nick Giannias, Presagis  
Patrick Lavoie, Presagis  
Richard Pitre, Presagis  
Roland Humphries, XPI Simulation  
Ryan Franz, FSI

#### **CDB Project Instigators**

Earl Miller, U.S. Government  
Jill Ashby, U.S. Government  
Joe Preston, U.S. Government  
Kevin Mobley, U.S. Government  
Larry Grice, U.S. Government  
Rita Simons, U.S. Government  
Victor Colon, U.S. Government