CDB X Conceptual Model with Prototyping Examples and Recommendations

Table of Contents

Publication Date: YYYY-MM-DD

Approval Date: YYYY-MM-DD

Submission Date: YYYY-MM-DD

Reference number of this document: <Something>

Category: Engineering Report from SOCOM/SOFWERX sponsored initiative; formatted like an OGC Public Engineering Report

Editors: David Graham, Carl Reed

Title: CDB X Conceptual Model with Prototyping Examples and Recommendations

---

Engineering Report from SOCOM/SOFWERX sponsored initiative; formatted like an OGC Public Engineering Report

WARNING

This document is not an OGC Standard. This document is an Engineering Report from SOCOM/SOFWERX sponsored initiative; formatted like an OGC Public Engineering Report created as a deliverable in an OGC Member (SOCOM) sponsored initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any Engineering Report from SOCOM/SOFWERX sponsored initiative; formatted like an OGC Public Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.

LICENSE AGREEMENT

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

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

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

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

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the

Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Chapter 1. Subject

This Engineering Report (ER) documents the results and recommendations of the rapid prototyping activites conducted during the 3D Geospatial Series Tech Sprint II - OGC CDB 2.0 (aka CDB X). This activity was performed in support of Special Operations Forces (SOF) Future Concepts. This effort hopes to accelerate evolution of the OGC CDB standard to meet the needs of planning, rehearsal, and Mission Command systems providing decision support to Special Operations Forces and enabling SOF tactical and operational advantage. OGC industry standards enable interoperability of geospatial data across systems and applications that SOF Operators and analysts use across warfighting functions.

Short summary of CDB X goal: Meeting the requirements for tactical GEOINT that can be used across warfighting functions.

Chapter 2. Executive Summary

Beginning in 2018 the CDB community has discussed and considered what CDB version 2.0 might might provide in terms of capabilities. These discussion also touched on architecture and related logical models. Many of the CDB 2.0 discussion occurred as part of the OGC standards work. More recently, other organizations such as SOCOM and NGA, have vigorously engaged in these discussions. However, progress has been relatively slow.

Therefore, in order to accelerate the development of CDB 2.0, in May 2020 SOFWERX announced the 3D Geospatial Series Tech Sprint II – OGC CDB 2.0 [https://events.sofwerx.org/3dgeots/]. In support of SOF Future Concepts, this effort hopes to accelerate evolution of the OGC CDB standard to meet the needs of planning, rehearsal, and Mission Command systems providing decision support to Special Operations Forces and enabling SOF tactical and operational advantage. Additionally, a requirement to better align aspects of the CDB standard with the One World Terrain activity was expressed during the initial workshop.

The CBD Tech Sprint was comprised of five (5) key work phases:

Phase 1: 22 June 2020 to 26 June 2020 Virtual Concept Development: In a series of virtual workshop conferences, participants discussed CDB 2.0 requirements and design objectives with a focus on defining conceptual and logical models for the CDB 2.0 standard. The discussion focused on five key topics: Attribution, 3D Models, Vector Data/GeoPackage, Coverage Tiling (and LoDs and general tiling), and Top Level Metadata. This models and related topics discussions served as the basis for technical experimentation in Phase 2.

Phase 2: 06 July 2020 to 31 July 2020 Virtual Tech Sprint: Participants, and/or their organizations, will provide prototype implementations of the conceptual model documented in Phase 1. Through technical integration experimentation, the model will be updated and refined.

Phase 3: 03 August 2020 to 28 August 2020 Engineering Report (ER): Participants will complete the project Engineering Report detailing the conceptual model and results of the technical integration experiments.

Phase 4: 07 September 2020 to 25 September 2020 Proposed Draft Specification: Participants will create a proposed draft specification for “CDB 2.0” based on work in the previous phases, to be submitted to the appropriate working groups within OGC, Khronos, and/or SISO.

NOTE	Once the experimentation activities started, the participants in concert with SOFWERX and SOCOM determined that writing a draft specification was beyond the scope of this project. Instead, the focus increased on the experiments and prototyping to provide key recommendations to be communicated in this ER and submitted to the OGC CDB Standards Working group (SWG) to provide the foundation for the actual standards development work.

Phase 5: 05 October 2020 to 5 November 2020 Final Report: Participants will write a final report summarizing activities to date, recommendations for the OGC SWG and other standards bodies to consider, and contributions of each participant.

In order to accomplish the target deliverables of this activity, the participants, based on their

expertize, were allocated to five focus groups:

1. 3D Subgroup
2. Attribution Subgroup
3. Coverages/Tiling Subgroup
4. Vectors in GeoPackage Subgroup
5. Metadata Subgroup

These teams focused on specific experiments and prototyping tasks to test key design concepts for inclusion in CDB-X. The results are documented in this ER.

NOTE	The CDB X Design Goals, Use Cases, and Conceptual/Logical Models chapter in this ER provides additional requirements and recommendations. These are phrased general design goals based on group discussions, design threads in the recommendations below, and discussions in the OGC CDB Standards Working Group.

2.1. Recommendations

The following are the key recommendations for changes, enhancements, and design principals for not just CDB-X but also CDB version 1.3. The version 1.3 changes are viewed as important enhancement that resolve both existing user specified needs as well as helping to provide a transition path for the future major CDB version. The following are the major recommendations. A complete summary of all Attribution recommendations can be found in Annex A.

NOTE	A key requirement for a CDB datastore is the ability to version content. This requirement was not addressed in any of the Subgroup experiments. However, all the Spring participants agreed that version management is a critical requirement for CDB-X. The CDB Versioning, old and new chapter of this ER provides details.

NOTE	A recurring discussion topic during the Sprint was a desire to better align the CDB Standard with the ongoing One World Terrain activity. This was a major determining factor in how the Attribution experiments and the use of GGDM were structured. This desire also helped direct the work of the 3D Modeling experiments. These experiments were successful.

2.1.1. 3D Subgroup

The following are the recommendations from the 3D Models and other 3D Content Subgroup.

`Recommendation 1`: The 3D model group recommends that a new CDB standard adopts the latest version of OpenFlight, leveraging the extended materials, hot spots and other important constructs it brings.

`Recommendation 2`: Complete the EXT_CDB draft glTF extension that adds required CDB simulation capabilities identified as missing from glTF. The new (and prototype) extension defines CDB-specific

constructs such as points, zones, lights, light maps, and extended texture types.

`Recommendation 3`: Consider possible standalone glTF extensions for: - Coordinate Reference System - Shared interest with One World Terrain and possibly wider glTF ecosystem. - Switch nodes - Shared interest with One World Terrain moving models format. Note: Node switching is supported in the I3S Community standard. - External references - Some interest in wider glTF ecosystem, but many unresolved implementation details.

`Recommendation 4`: Leverage existing glTF extensions: - Articulations - This extension adds articulations, attach points, and related metadata to the glTF model (see AGI_articulations). - Metadata - Consider the EXT_feature_metadata extension. This extension is for storing metadata at different levels of granularity including per-mesh, per-face, per-vertex, and per-texel. This could be a good candidate for CDB model attributes and material map textures.

`Recommendation 5`: Consider a Model indexing file that includes the following: Model LODs, Interior and exteriors, Navigation and collision mesh

`Recommendation 6`: To reduce the current number of files in CDB 1.x a recommendation is to group each model data into a zip (all LODs, textures etc…) while keeping the xml (index file) separate. A quick access to the index file would allow identifying what is required to be loaded into each ZIP. 9See Recommendation 5). This is a possible CDB 1.3 enhancement.

2.1.2. Attribution Subgroup

The following are the major recommendations from the Attribution Subgroup. Each of the major recommendations contain multiple "minor" recommendations that result from considering the implication of the major recommendation. The complete list of major/minor recommendations for Attribution are found in both the Attribution section and Annex A.

`Recommendation 1`: The subgroup recommends extending the current CDB XML metadata to add the NAS metamodel capabilities that are currently not supported.

`Recommendation 2 and design goal`: The CDB core conceptual model should not mandate any particular data dictionary or content, but rather provide the conceptual and logical metamodel for describing any ISO 19109 compliant application schema to the maximum extent practical. There should be no technical reason why one could not develop an extension profile for CDB for any particular data dictionary that complies with ISO 19109.

`Recommendation 3`: Adopt NAS-compliant logical entity-attribute model for CDB X with extensions for CDB use cases.

`Recommendation 4`: Delegate entity and attribute physical encoding choices to vector and 3D model containers instead of specifying globally.

`Recommendation 5`: Define backward-compatible extensions in CDB 1.3 to add constructs necessary to move toward NAS-compliant attribution

The following are recommendations for possible inclusion in CDB version 1.3.

`Version 1.3 Recommendation - Extended Attributes` The subgroup discussion on this topic is titled:

Should Extended Attributes be preserved at the logical data model level? [https://github.com/sofwerx/cdb2-concept/issues/25] The suggestion is that the CDB SWG discuss this issue and possible solution as a possible change for CDB version 1.3. Some additional testing may be required to determine if this capability can be added to version 1.3 or not.

`Version 1.3 Recommendation - Attribute default values`: The subgroup discussion on this topic is titled: Attribute Default Values #32 [https://github.com/sofwerx/cdb2-concept/issues/32]. The recommendation is that Defaults.xml can be used to define global attribute defaults as well as per-dataset defaults. Doing per-entity defaults would be a straight forward extension that could be proposed for CDB 1.3 as a transition path. The subgroup suggests that the CDB SWG discussion this for possible inclusion in version 1.3. A change request for this approach to specifying default values is also suggested.

`Version 1.3 Recommendation - Attribute Terms` The subgroup discussion on this topic is titled: Capture Attribute Terms (Enumerants) in Metadata #31 [https://github.com/sofwerx/cdb2-concept/issues/31]. Attributes describing qualitative values are present in CDB 1.2 and the list of valid values for each attribute are documented in the human-readable specification with both the vocabulary term name and its integer numeric value (index). However, the machine-readable XML metadata does not contain any of this information and treats these attribute types as raw integers with only a minimum and maximum value constraint. It may make sense as a transition path to update CDB 1.3 to define additional XML elements in a backward compatible way to capture these definitions from the existing specification into the machine-readable XML metadata. The conceptual model in the CDB 1.2 specification does align with how GGDM treats such attributes, so there is no fundamental incompatibility, and the proposed CDB X dictionary design accounts for properly tracking the terms for qualitative attributes in a machine-readable way in SQLite.

2.1.3. Coverages/Tiling Subgroup

The following are the recommendations from the Coverages/Tiling Subgroup.

`Recommendation 1`: Any tiling schemes specified in a CDB X data store (repository) SHALL be based on and consistent with the: OGC Core Tiling Conceptual and Logical Models for 2D Euclidean Space (19-014r3) and OGC Two Dimensional Tile Matrix Set Standard (17-083r2)

`Recommendation 2`: LoD Grouping - For users at the edge and smaller areas, that all the CDB-X coverage layers be present within a single GeoPackage container.

`Recommendation 3`: LoD Grouping - For Modeling and Simulation uses as well as data repository cases, that a series of GeoPackage containers be used to store CDB X coverage layers.

`Recommendation 4`: Define the capability for splitting GeoPackages based on a specific tiling scheme outside of the CDB X standard so that this split content can be used by itself as a component of other non-CDB based applications.

`Recommendation 5`: Use the proposed cdb.json index of packages and data layers. This would allow defining the description of the packages and LOD grouping outside of the CDB-XX standard so that description can be used elsewhere as well.

`Recommendation 6`: Elevation min-max - Move the minimum and maximum elevation values for the gridded elevation coverage contained in a tile to the tile metadata.

Recommendation 7: Image Compression - That loss-less and lossy image compression solutions be explored for use in CDB-X. Any such solutions are not viewed as a replacement for JPEG 2000 but instead as alternatives.

Recommendation 8: Materials - CDB-X needs to support material data to provide the same functionality as CDB 1.x. To also reduce the number of files, this can be accomplished by putting all the raster material data (including material table) in a single CDB data layer in GeoPackage, perhaps using the GeoPackage Related Tables Extension.

Recommendation 9: Although the use of non-tiled vector data layers (e.g. storing the geometry as WKB in GeoPackage features tables) should also be specified in the CDB Standard, the use of a tiled vector data extension should also be allowed. In particular, tiling vector data is essential for dealing with features spanning a very large geospatial extent, such as coastlines.

Recommendation 10: GeoPackage. The imagery in these GeoPackages is lossy. Therefore, allow the use of JPEG-2000 and/or additional lossless formats more compact than PNG in GeoPackages. This should be submitted as a change request to the GeoPackage Standards Working Group.

Recommendation note: Supporting more than one tiling scheme in a version of CDB is not recommended. The choice of the tiling scheme is foundational to how data layers are processed and stored and accessed.

2.1.4. Metadata Subgroup

The Metadata chapter in this ER provides the following guidance and recommendations:

Recommendation 1: All Sprint participants agreed that metadata including provenance is a critical requirement for the CDB-X Standard. They also agreed that some elements should be mandatory.

Recommendation 2: Metadata and provenance content should be self-describing.

Recommendation 3: Keep the core set of mandatory metadata elements limited and simple. Collecting and maintaining metadata can be costly – unless workflows are designed to capture metadata as part of the production process.

Recommendation 4: Define an extensible CDB metadata model that allows for easily incorporating additional metadata elements for specific data types, domains or applications. A good example would be the metadata required to make SWIR and NIR in a CDB data store useful by discovery, access, processing, and visualization services for those data types.

Recommendation 5: Discuss and agree on element names for the mandatory elements. This is because each metadata standard names elements differently. This also suggests that a metadata element crosswalk document may be required. The beginnings of such a document were developed as part of the CDB 1.1 revision work.

Recommendation 6: Every CDB dataset should have its own metadata that describes the content of that specific dataset. This will allow for much more flexible extensibility of new data types, enhances the value of existing datasets and enhances discoverability.

Recommendation 7: Consider whether the GeoPackage Metadata extension is robust and flexible

enough to meet CDB-X requirements.

2.1.5. Vectors in GeoPackage Subgroup

The following are the key recommendations from the Vectors in GeoPackage Subgroup>>.

`Recommendation 1`: Storing large numbers of feature data in single GeoPackage containers and retrieving that data by applying spatial and attribution filters that correspond with typical CDB access patterns appears to be practical. Therefore, the CDB-X core standard should specify requirements that support storing all vector data in a single GeoPackage.

`Recommendation 2`: Spatial filters appear to easily mimic the existing CDB tiling scheme. Therefore, the CDB-X core standard should specify requirements that support the ability to 1.) Specify such filters and 2.) Provide highly performant queries.

`Recommendation 3`: Storing 'significant size' on model instancing point features can significantly improve the model retrieval scheme, rather than storing models in the significant size related folder scheme. Storing and evaluating significant size on instancing points can make visual content and performance tuning much more practical.

2.2. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Contacts

Name	Organization	Role
David Graham	Eaglecapsystems	Editor
Carl Reed, PhD	Carl Reed & Associates	Editor
Kevin Bentley	Cognitics	Contributor
Holly Black	CAE	Contributor
Hermann Bressard	Presagis	Contributor
Patrick Cozzi	CESIUM	Contributor
Brian Ford	FlightSafety	Contributor
Ryan Franz	FlightSafety	Contributor
Jay Freeman	CAE	Contributor
Jérôme Jacovella-St-Louis	Ecere	Contributor
Michala Hill	Cognitics	Facilitator/Contributor
Greg Peele	Geometric Progress	Contributor
Vaughn Whisker	ARL PSU	Contributor
Tracey Birch	CloudLake/USSOCOM SOF AT&L	Emeritus

Chapter 3. References

The following normative documents are referenced in this document.

NOTE: Only normative standards are referenced here, e.g. OGC, ISO or other SDO standards. All other references are listed in the bibliography.

- OGC: 15-113r5 Volume 1: CDB Core Standard: Model and Physical Data Store Structure [https://portal.opengeospatial.org/files/15-113r5]
- OGC: 16-007r4 Volume 11: CDB Core Standard Conceptual Model [https://portal.opengeospatial.org/files/16-007r4]
- OGC: 12-128r15 Geopackage Encoding Standard - with Corrigendum version 1.2.1 [https://portal.opengeospatial.org/files/12-128r15]
- OGC: 18-000 GeoPackage Related Tables Extension [http://docs.opengeospatial.org/is/18-000/18-000.html]
- OGC: 17-066r1 GeoPackage Extension for Tiled Gridded Coverage Data [http://docs.opengeospatial.org/is/17-066r1/17-066r1.html]
- Khronos GL Transmission Format (glTF) version 2.0 [https://github.com/KhronosGroup/glTF/tree/master/specification/2.0]

Chapter 4. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard OGC 06-121r9 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2] shall apply. In addition, the following terms and definitions apply.

4.x
authoritative data
Officially recognized data that can be certified and is provided by an authoritative source.

(SOURCE: Authority and Authoritative Sources: Clarification of Terms and Concepts for Cadastral Data. FGDC/NIST 2008)

4.x
authoritative data source
An information technology (IT) term used by system designers to identify a system process that assures the veracity of data sources. These IT processes should be followed by all geospatial data providers. The data may be original or it may come from one or more external sources all of which are validated for quality and accuracy.

(SOURCE: Authority and Authoritative Sources: Clarification of Terms and Concepts for Cadastral Data. FGDC/NIST 2008)

4.x
conceptual model
description of common concepts and their relationships, particularly in order to facilitate exchange of information between parties within a specific domain. A conceptual model is explicitly chosen to be independent of design or implementation concerns.

(SOURCE: CEN ENV 1613:1995)

4.x
coverage
feature that acts as a function to return values from its range for any direct position within its spatio-temporal domain

4.x
coordinate reference system
coordinate system that is related to the real world by a datum

(SOURCE: ISO 19111:2019 Geographic information — Referencing by coordinates)

4.x
dataset
A dataset is a collection of data, published or curated by a single agent. Data comes in many forms including numbers, words, pixels, imagery, sound and other multi-media, and potentially other types, any of which might be collected into a dataset.

Note: There is an important distinction between a dataset as an abstract idea and a distribution as a manifestation of the dataset

(SOURCE: W3C Data Catalog Vocabulary (DCAT) - Version 2 [https://www.w3.org/TR/vocab-dcat-2/], 2020)

4.x
Data Store

A data store is a repository for persistently storing and managing collections of data which include not just repositories like databases, but also simpler store types such as simple files, metadata, models, etc.

(SOURCE: https://www.information-management.com/glossary/d.html:2020)

4.x
elevation
Synonym for “height”

(SOURCE: OGC Abstract Topic 6: Schema for coverage geometry and functions [https://portal.opengeospatial.org/files/?artifact_id=19820])

4.x
feature data dictionary
A Feature data dictionary is a collection of descriptions of the Feature objects or items in a Feature data model for the benefit of programmers and others who need to refer to them.

(SOURCE: OGC Testbed-13: CDB Engineering Report - 17-042)

4.x
foundation data
authoritative data and models that have been curated and stored in a CDB repository. Foundation data uses known and agreed to controlled vocabularies for attribution, has been curated based on the requirements as stated in the CDB standard, and supported by the required metadata.

4.x
geospecific model
A Geospecific model 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 features. NOTE: Being discussed for possible evolution as part of CDB 2.0

4.x
geotypical model
A Geotypical model is instanced multiple times within a CDB data store. Geotypical models correspond to representative (in shape, size, texture, materials and attribution) models of real-world manmade or natural 3D features.

4.x
height
Distance of a point from a chosen reference surface measured upward along a line perpendicular to that surface. Note 1 to entry: A height below the reference surface will have a negative value, which would embrace both gravity-related heights and ellipsoidal heights.

(SOURCE: ISO 19111:2019 Geographic information — Referencing by Coordinates)

4.x
Metadata
information that captures the characteristics of a resource to represent the ‘who’, ‘what’, ‘when’, ‘where’, ‘why’, and ‘how’ of that resource.

(SOURCE: National System for Geospatial Intelligence (NSG) Metadata Foundation (NMF) Version 3.0 [https://nsgreg.nga.mil/doc/view?i=4252&month=10&day=22&year=2019])

4.x
repository

a place that holds CDB data, makes CDB data available to use, and organizes CDB data in a logical manner. (Network of the National Library of Medicine) 2020

4.x
resource
identifiable asset or means that fulfils a requirement

NOTE	A web resource, or simply resource, is any identifiable thing, whether digital, physical, or abstract.

(SOURCE: ISO:19115-1:2014 Geographic information — Metadata — Part 1: Fundamentals)

- **tile**

  geometric shape with known properties that may or may not be the result of a tiling (tessellation)process. A tile consists of a single connected "piece" without "holes" or "lines" (topological disc).

NOTE	“tile” is NOT a packaged blob of data to download in a chunky streaming optimization scheme!

- **tiling**

  in mathematics, a tiling (tessellation) is a collection of subsets of the space being tiled, i.e. tiles thatcover the space without gaps or overlaps.

4.1. Abbreviated terms

AGC	US Army Geospatial Center
AIXM	Aeronautical Information Exchange Model
ATAK	Android Tactical Assault Kit
COTS	Commercial Off The Shelf
CRS	Coordinate Reference System
DIGEST	Digital Geographic Information Exchange Standard
DGIM	Disaster Geo-Information Management

DGIWG	Defence Geospatial Information Working Group
EDCS	Environmental Data Coding Specification
GGDM	Ground-Warfighter Geospatial Data Model
GPKG	GeoPackage
glTF	GL Transmission Format
FACC	Feature Attribute and Coding catalog
FDD	Feature Data Dictionary
FSC	Feature Sub-code
LoD	Level of Detail
MC	Mission Command
NAS	NSG Application Schema
NCV	NSG Core Vocabulary
NFDD	National Feature Data Dictionary
NGA	National Geospatial Intelligence Agency
NSG	National System for Geospatial-Intelligence
OTW	Out the Window
OWT	One World Terrain
PBR	Physically-Based Rendering (PBR)
SOF	Special Operations Forces
STAC	SpatioTemporal Asset Catalog
STE	Synthetic Training Environment
TIFF	Tagged Image File Format
TMS	Tile Map Service
TMS	Tile Matrix Set
UML	Unified Modelling Language

Chapter 5. Overview

Section 5 CDB 2.0 - Background and History: Provides the background and history of CDB to provide the context for the CDB-X work described in this Engineering Report.

Section 6 CDB X Design Goals, Use Cases, and Conceptual/Logical Models: Documents any models, such as the OGC/ISO Simple Features geometry model, that are germane to the CDB-X work.

Section 7 Attribution in CDB-X: Section 7 discusses the Attribution Topic and the work and recommendations of the Attribution Subgroup.

Section 8 CDB Vector Data in Large GeoPackage Containers: Documents experiments to determine the viability of storing very large volumes of vector data in a smaller number of large GeoPackage files rather than a very large number of small Shapefiles.

Section 9 3D Models and Other 3D Content: Discusses the proposal to use glTF to encode and store 3D models in the CDB-XX data store

Section 10 CDB X Metadata and Integration: This section discusses metadata in CDB-X.

Section 11 Tiling and packaging of data layers: This section evaluates and discusses approaches to packaging one or more data layers in GeoPackages (including coverage, vector and 3D data) for CDB-X.

Chapter 6. CDB 2.0 - Background and History

This section of the ER provides background information on the history and development of version 1 of the CDB standard. This background information provides the basis for the decision for the SOFWERX/SOCOM sponsored CDB X Conceptual Modelling activity.

6.1. What is CDB?

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

The application of CDB to future simulation architectures will significantly reduce runtime-source level and algorithmic correlation errors, while reducing development, update and configuration management timelines. With the addition of the High Level Architecture [https://en.wikipedia.org/wiki/High-level_architecture] - Federation Object Model (HLA/FOM) and Distributed Interactive Simulation [https://en.wikipedia.org/wiki/Distributed_Interactive_Simulation] (DIS) protocols, the application of the CDB standard provides a common environment to which inter-connected simulators share a common view of the simulated environment.

The CDB standard defines an open format for the storage, access and modification of a synthetic environment database. A **synthetic environment** is a computer simulation [https://en.wikipedia.org/wiki/Computer_simulation] that represents activities at a high level of realism, from simulation of theaters of war to factories and manufacturing processes. These environments may be created within a single computer or a vast distributed network connected by local and wide area networks and augmented by super-realistic special effects and accurate behavioral models. SE allows visualization of and immersion into the environment being simulated see "Department of Defense Modeling and Simulation (M&S) Glossary", DoD 5000.59-M, [https://www.msco.mil/MSReferences/Glossary/MSGlossary.aspx].

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

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

attributes of the synthetic environment data as well as their relationships.

6.2. OGC CDB 1.x History

The CDB specification was initially authored by CAE Inc. on November 2005 under a contract administered by the US Army Program Executive Office for Simulation Training and Instrumentation (PEO STRI) to meet requirements set by US Special Operations Command (USSOCOM)for interoperable, high-performance mission rehearsal and simulation federations. CAE along withother companies (e.g., Flight Safety Inc., New York, NY, USA, Presagis Inc., Montréal, QC, Canada, etc.) have delivered hundreds of simulation channels based on CDB to customers in the US, Canada, UK, Germany, Turkey, Israel, Singapore, Australia, Brunei and other countries.

To improve the efficiency of M&S analysis, facilitate data and services' interaction and communications, and reduce operational cost and complexity, the CDB Community led by CAE submitted the industry specification version 3.2 into the OGC for consideration as an OGC Best Prctice. Beginning in 2013, CDB was discussed and demonstrated at OGC Technical Committee meetings. The industry specification was approved by the OGC Members and released as an OGC Best Practice [https://portal.opengeospatial.org/files/?artifact_id=61935]. The OGC CDB SWG then split the two Best Practice documents into 12 volumes, removed unnecessary content by providing links to authoritative sources, and began evolving terminology to be consistent with OGC/ISO terms and definitions. CDB Version 1.0 was approved by the OGC membership in October 2016 as a new standard.

The CDB SWG continued the evolution of CDB. Version 1.1 of the CDB Standard and related Best Practices was approved in August 2018. One of the main enhancements for version 1.1 was guidance on how to encode and include global and local spatial metadata in a CDB data store. Version 1.1 also moved to having file extensions being generic in all CDB Requirements clauses. This was done to allow inclusion of new (additional) file types in a CDB data store. Further, a number of CDB User communitu chnages requests were included in version 1.1.

Version 1.2 of the CDB Standard was approved as an official OGC Standard in August 2020. The major enhancement in CDB version 1.2 was the inclusion of the ability to add OGC GeoPackage formated containers into a CDB data store. There are two new CDB volumes that explicitly provide requirements and guidance on using GeoPackages in a CDB data store. Version 1.2 also includes two substantive changes. First, the way that CDB's Primary Alternate Terrain Elevation dataset was defined in CDB Version 1.1 and earlier causes problems with standard open source libraries used to read and process these data. The SWG agreed to changes address the issues that ground simulation has with CDB gridded terrain meshes. The second substantive change is that CDB 1.1 and earlier supported a single (hard-coded) file format per dataset. To allow other fileformats to be used in a CDB Data Store, the need to explicitly specify the file format that is used to physically store the components of a given dataset is required.The current CDB metadata and controlled vocabulary definitions (See Clauses 1.4.3, 3.1, and 5.1 inCDB Volume 1: Core) has a file called Datasets.xml listing all possible datasets that can be used in aCDB data store. The file has been expanded to indicate the encoding format used to encode the dataset and its components.

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

environmental richness, quality and resolution.

6.3. OGC CDB 2.0

Beginning in late 2020, the OGC CDB SWG began discussion on what CDB version 2.0 might have as an architecture, what additional encodings and.or formats might be supported, and so forth. These discussion also included initial thoughts on a CDB 2.0 conceptual model. However, progress was slow - in part due to other OGC Member commitments and resource constraints.

NOTE	In the OGC, a major version number indicates that the new version is not backwards compatible with previous versions of the standard. The use of major revisions allows for a major evolution and enhancement to any OGC Standard.

A number of the major requirements for version 2.0 have been discussed for some time and have been captured in various OGC documents including formal Change Requests and OGC Interoperability Program Engineering Reports.

Some of the major enhancements that have been discussed and documented are:

1. Develop an approach (and model?) for using any attribution controlled vocabulary in a CDB data store.
2. Determine if the current tiling scheme is optimal or should be enhanced/replaced with a better tiling approach.
3. Describe explicitly how the CDB model may or may not align with the OGC DGGS standard;
4. Provide best practice details on how to use others OGC standards such as WMS and WFS as well as they new OGC APIs to access existing CDB data stores. This work may require Interoperability Experiments to better understand the implications of these decisions;
5. Extend the supported encodings and formats for a CDB structured data store to include the use of the CityGML, and InDoorGML standards as well as other broadly used community encoding standards, such as glTF and GeoTIFF. This work may require performing OGC Interoperability Experiments to better understand the implications of these decisions.
6. Further align CDB terminology to be fully consistent with OGC/ISO terminology.

Making these enhancements may allow the use and implementation of a CDB structured data store for application areas in addition to aviation simulators.

6.4. Attribution - An OGC Testbed 13 CDB Activity in

The CDB Engineering Report [http://docs.opengeospatial.org/per/17-042.html] (ER) summarizes the CDB sub-thread activity. The CDB activity was divided into three main work packages:

- A feasibility study;
- The implementation of data models and schemas mapping that are based on the feasibility study results; and
- A set of OGC web services that implement the CDB in the form of WFS and WCS (Web Coverage

Service) instances.

This Testbed 13 Engineering Report describes the approach taken and the results of the experimentation:

- The conceptual model of an OGC CDB 1.0 datastore as a UML (Unified Modeling Language) diagram to show different datasets (the 3D models, vector features and coverages) structure;
- How to process and use a NAS-based Profile as a CDB feature/attribute data model or a GML-SF0 application schema;
- How to access, navigate and visualize a CDB dataset using OGC web services (such as WFS and WCS).

The Testbed 13 CDB activity also resulted in a formal OGC Change Request Proposal [http://ogc.standardstracker.org/show_request.cgi?id=544] that was submitted for consideration in April 2018. The Change Request provides:

- Recommendations for replacing FACC feature code and indexing structure to be consistent with the application schemas (e.g. NAS data model) under discussion for the OGC CDB 2.0;
- Recommendations for supporting application schemas in CDB (level of complexity: Esri Geodatabase, GML-Simple Features Level 0 application schema) are being discussed for the OGC CDB 2.0;
- A method to expand the supported encodings and formats for an OGC CDB compliant datastore;
- Guidance on generating a coherent attribute schema for CDB 1.0 based on the "CDB_Attribute.xml" file.

Chapter 7. CDB X Design Goals, Use Cases, and Conceptual/Logical Models

This section documents draft design objectives, the target use cases, and the CDB X Conceptual and Logical Models. This section is currently a work in progress.

NOTE	Some of the content in this section is paraphrased from the recently available USGIF/OGC paper, "Advancing the Interoperability of Geospatial Intelligence Tradecraft with 3D Modeling, Simulation and Game Engines."

7.1. Use Cases

The following are summaries of key use cases identified during this project.

`Tactical Use Case - CDB-T`: Support tactical forces with CDB content for both planning and deployment. There is also the consideration of using the content on mobile devices for the Warfighter at the Edge [http://docs.opengeospatial.org/per/19-030r1.html]. Possible impact: Define a profile of the foundation data in a form most suitable for tactical devices (CDB-T, where T is for tactical), such as ATAK. This is the CDB-T use case where "T" is for "Tactical".

`Gaming Use Case - CDB-G`: More and more gaming systems, such as Unity, are depending on "real" 2D and 3D geospatial content to add realism to their gaming systems. This is because gaming technology is supporting the visualization, content generation and hybrid-streaming of 3D geospatial data such as combining optimized performant content with dynamic geospatial events and condition. Possible impact: Define a derivative and optional structure of the foundation data in a form most suitable for gaming devices. This is the CDB-G use case where "G" is for "Gaming"

`M&S Use Case - CDB-S`: A CDB datastore needs to provide a modeled environment representation for distributed simulation applications. This representation needs to be currated, authoritative, and as up to date as possible. Possible impact: Define a porfile of the foundation data in a form most suitable for simulators. (CDB-S, where S is for simulators)

`Analytics Use Case - CDB-A`: Increasingly there is a need to access and use content available in a CDB datastore for analytics such as change detection, mobility analysis, and integration with other models. This leads to consideration of new approaches and methods for integrating 3D analytics with mission planning and mission rehearsal synthetic training environments such as the inclusion of ground-clutter in urban environments. Possible impact: Define a derivative and optional structure of the datasets in a form most suitable for analytics using AI/ML. This is termed `CDB-A`, where A is for analytics)

`Data Transport Use Case` <need short description> Possible impact: Optimize the speed data synchronization between all echelons and data systems for CDB-F and CDB-T

`DDIL Deployment Use Case` CDB-F, CDB-T, and CDB-A must be able to deploy in DDIL environments. Also see CDF-Tactical above

NOTE	Optimization For Use Cases: CDB-S, CDB-G and CDB-T may tile/generate LODs and derivative data types (assuming they are OGC and/or Khronos standards) in a manner to optimize efficiency and performance leveraging a small, finite set of tiling schemes (Profile should include Tiling structure, LOD structure, optimization, etc.) CDB-A may generate topology, extended attributes and derivative data types (assuming they are OGC and/or Khronos standards) in a manner to optimize efficiency and performance leveraging a small, finite set of relationship schemes

This is future work and actually would not be part of a core CDB-X standard Configuration Management. Define an abstract mechanism for configuration management applicable to multiple deployment strategies (file systems, RDMS and cloud) relational databases for CDB-F, CDB-S, CDB-G, CDB-T and CDB-A. Support Versioning, attribution, synchronization

7.2. CDB X Design Goals and Objectives

The following are the design goals and objectives for the development of CDB-X.

- Be as performant as possible in terms of 1.) Speed of access and 2.) Impact on hardware resources.
- Consider how to incorporate identified recommendations into the next minnor CDB revision (1.3) to
  - Enhance current capabilities,
  - Not break backwards compatibility,
  - Provide a more seamless migration path moving to CDB-X.
- Geometry model: All geometry for any vector feature data shall be compliant with the OGC/ISO Simple Features Geometry Model.
- Attribute model: CDB-X shall be consistent with the NAS/GGDM and harmonized as approporiate the current CDB attribution model.
- CDB-X Standards Sructure: CDB-X should evolve the standard to a "Core" that specifies a set of requirements that all CDB extensions, profiles, and profiles with extensions shall conform to. Other characteristics of the Core are:
  - The core shall support the tiling of any layer as does the current CDB standard.
  - The core shall also specify that vector data does not need to be tiled.
  - The core shall support specification for LoDs.
  - The core shall be implementable in a file system (as today) or in a container technology such as GeoPackage.
  - The core shall be implementable in the cloud.
- Any profiles based on the core shall have the following requirements:
  - Correlation Requirement: CDB-T, CDB-S, CDB-G and CDB-A profiles shall correlate to the data defined in CDB-F
  - Derivative and Optional structures: CDB-T, CDB-S, CDB-G and CDB-A profiles shall encode its rules and mechanisms for optimization in a self-describing form.

    - Holistic Foundation: CDB-T, CDB-S, CDB-G and CDB-A structures shall all be generated from the data defined in CDB-F
- 3D Graphics Models - CDB-X shall support additional model encoding beyond OpenFlight. glTF shall also be supported.
- Metadata - CDB-X shall support mandatory and optional metadata elements including support for provenance. Any profiles shall clearly specify metadata elements/requirements for that profile.
- The ability to version the contents of a CDB datastore is mandatory. The requirement is discussed in detail in the Versioning section of this ER.

7.3. Concepts and Overarching Requirements

The following section provides additional background on key concepts and/or overarcing requirements germane to the definition and development of a CDB-X Standard.

7.3.1. Interoperability

The Interoperability of software, workflows and data across the GEOINT and M&S communities is required to improve mission responsiveness by leveraging a growing diversity of geospatial information and just in time fusion of information from drones, body cameras, phones and other IoT sensors. Interoperability of all the formats, encodings, and models used in CDB-X will enhance the ability to discover, access, update, and use content in a CDB datastore to meet the needs of the use cases identifed above.

7.3.2. What is Foundation Data

Foundation data are the authoritative data and models that have been curated per the rules defined in the CDB standard, and stored in a CDB repository. Foundation data utilizes known and consensus controlled vocabularies for attribution, has been curated based on the requirements as stated in the CDB standard, and the required metadata has been documented.

For example, as per the current CDB standard (and industry best practice), a road network (RoadNetwork dataset in CDB 1.2) could be considered as foundation data. In CDB 1.2, there are a set of requirements associated with how the roads data are structured and attributed and stored in a CDB repository.

The specification of any foundation data layer (such as RoadNetwork) at the logical model level can be 100% independent from the storage environment, end user use case(s) and so on while at the same time also having the requirements based on industry knowledge and use case requirements.

For example:

Requirement: The RoadNetwork shall be considered as a foundation data layer in a CDB repository.

Sub-requirement: All roads in a roads layer SHALL use a known road classification system to identify the roads type. This could be the US DoT classification, the current FACC classification, or the newer GDDM.

Sub-requirement: The RoadNetwork SHALL be topologically structured. (This enables network analysis etc). NOTE: The how is not specified at the logical model level.

Sub-requirements: The RoadNetwork Dataset SHALL be used to specify all of the road networks..

Notice that there is no mention of the actual storage structure, enabling software, storage formats, tiling and so forth.

7.3.3. Reference Systems

7.4. 3D Models

7.5. Attribution

7.6. Tiling/Coverages/Imagery

7.7. Vector Data

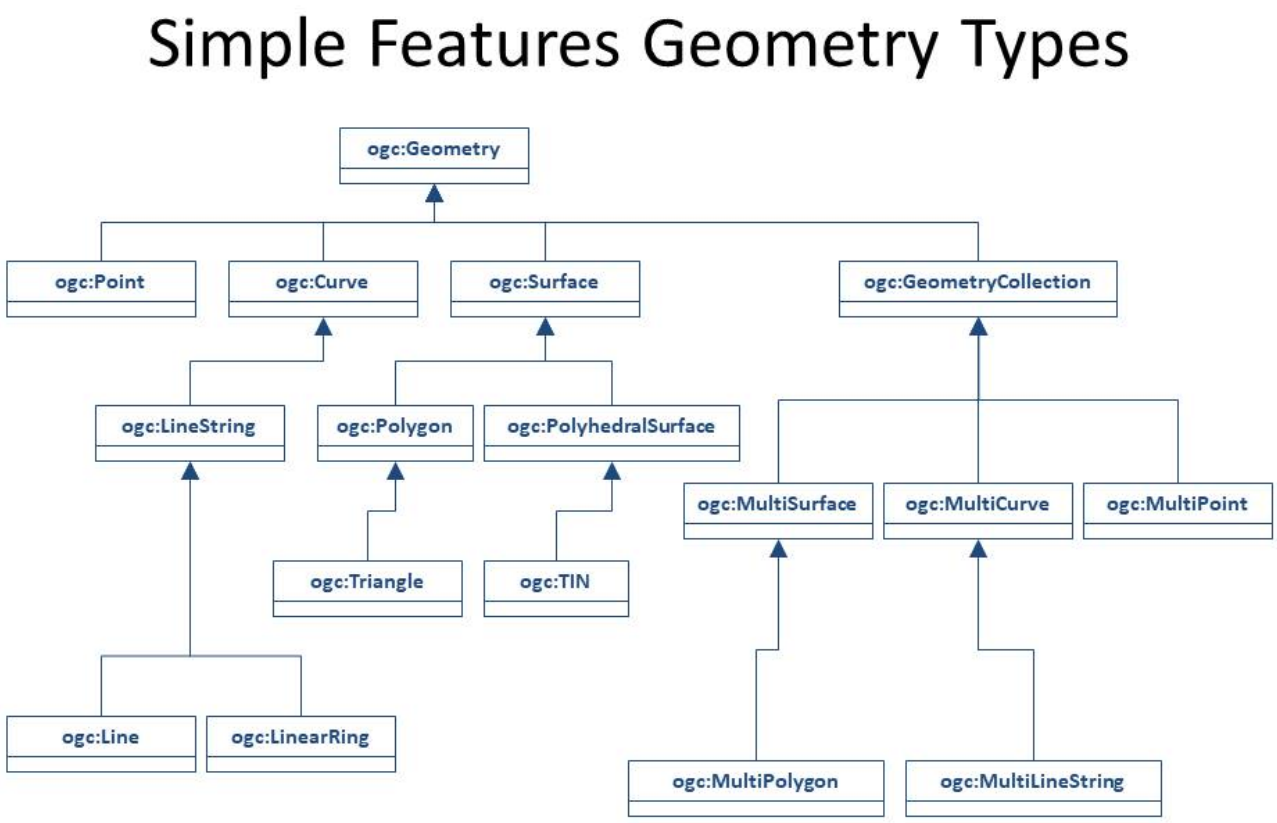


Figure 1. Simple Features Geometry Model.

Chapter 8. CDB-X Attribution

8.1. Overview of the CDB-X Attribution Work

NOTE This content of this section was authored by Greg Peele, leader of the Attributes Subgroup of the CDB-X activity. For purposes of this section, the Attribute Subgroup is hereafter referenced simply as the `subgroup`.

CDB stakeholders have stated the desire to upgrade OGC CDB to modern attribution standards. The current OGC CDB standard relies on extending the obsolete Feature Attribute Coding Catalogue (FACC) to define semantics of features and attributes. Over the 20 years since the last version of FACC, substantial work has been done on UML-based feature and attribute data models framed as ISO-compliant application schemas. Reviewing the adoption of the NSG Application Schema (NAS) via the Ground-Warfighter Geospatial Data Model (GGDM) for the US Army's One World Terrain, the CDB-X initial sprint identified that a similar approach could be a path forward for CDB and formed the Attribute Subgroup to evaluate the idea. The subgroup created a notional diagram for representing a NAS-compliant feature dictionary. The subgroup also proposed a storage design that consolidated the CDB feature dictionary metadata, currently spread across several XML files, into a single SQLite file for easier runtime use.

The subgroup then planned and executed three experiments to assess the feasibility of migrating CDB to NAS/GGDM and what mismatches or gaps existed. The first experiment created a prototype software code library for the notional feature dictionary design. This prototype defined runtime classes and code to populate them from existing OGC CDB 1.1 metadata, from the official GGDM 3.0 Excel sheet, and the notional CDB-X SQLite metadata. The code experiment was used to identify gaps in the feature model and mismatches in the conceptual model between current CDB and NAS/GGDM. The second experiment mapped the current CDB feature dictionary to NGA's Topographic Data Store (TDS) and GGDM and assessed the completeness and complexity of the mapping. The third experiment was a thought experiment to identify which changes to the feature dictionary would have ripple effects that changed the CDB data storage organization or content.

The experiment results showed that migrating CDB to a NAS-compliant data model via GGDM is feasible. Most changes are doable as incremental backwards-compatible updates to the current CDB XML metadata along with deprecating obsolete items, with no breaking impacts on data storage organization or content. The software prototype showed the CDB feature model to be mostly a subset of NAS, although not all of the OGC CDB specification is captured in the XML metadata now. CDB Feature Subcodes will need to be removed since no similar concept exists in NAS, and feature types identified that way will need to be mapped to new feature types or using enumerated attributes. Attributes will need to be specified at the feature level instead of at the dataset level. Existing constraints for attributes - particularly listing enumerated values - must be captured in the CDB metadata along with adding constraints such as text patterns and text-based enumerated codelists. We also proposed backward-compatible approaches for NAS multi-valued attributes, range-valued attributes, complex attributes with subfields, and feature-level relationships that gracefully degrade on existing CDB clients. Feature-level relationships can also implement feature metadata and shared attribution. With these extensions, we can create a new feature dictionary for GGDM and NAS in the current CDB format. The ripple effect thought experiment identifed that this does not impact the vector encoding and the only impact to the 3D model encoding is that changes

to feature types and codes will change model filenames.

The mapping experiment found that 30% of the CDB feature definitions can be migrated to GGDM easily. The gaps primarily arise from GGDM being a subset of NAS for outdoor environments intended for ground combat, so we believe the current NAS will fill most gaps in maritime and aeronautics feature types. A full CDB to NAS mapping should be created as a follow-up experiment. Any remaining gaps might be filled by parallel DGIWG work or by domain-specific standards like S-100 and AIXM. Two major gaps not addressed by NAS are building interiors (including underground content) and detailed vegetation. Substantial effort will be needed to reconcile CDB building interiors with existing incompatible standards like OGC CityGML, Industry Foundation Classes (IFC), Apple Indoor Data Mapping Format (IMDF), and so on. Effort will also be needed to synthesize a vegetation data model from environmental agency datasets. Conceptually, the CDB materials, light names, and mdoel components should also be aligned with the data dictionary rather than being entirely separate concepts, and future work will be needed to achieve that. We anticipate substantial overlap with One World Terrain on most of these extensions.

8.2. Background, History, and the Path Forward

One of the strengths of CDB as an interoperability standard, beyond the underlying geospatial representation standards it deploys, is the adoption of a controlled semantic vocabulary for feature types, attributes, and values. This approach - variously called a data model, a feature dictionary, or an entity catalog - is a necessary component of semantic interoperability between simulations. The feature attribution model goes beyond specifying the physical geometry, spatial referencing, and file formats used to represent the CDB data store to ensure simulations have a consistent and correlated deep understanding of the content semantics - to ensure a road is always a road, a lake is distinct from a river, width and height always has a consistent meaning, and so on. This semantic interoperability is necessary to ensure proper operation of simulation behaviors both for supporting human players and artificial intelligence agents.

The current CDB standard for attribution is primarily based on the officially obsolete DIGEST Feature Attribute Coding Catalogue (FACC) standard and has not integrated ongoing development and standards developed by other activities since OGC CDB 1.0. For several years, there have been discussions concerning how to modify the CDB standard to support newer feature attribution coding schemes and controlled vocabularies to align with current standards and best practices. While the existing CDB standard theoretically supports representing other controlled vocabularies through modifying the CDB metadata XML (with some gaps identified in this report), at this time no changes to the CDB standard to adopt or promote other coding schemes has occured. This section provides 1.) information on the current approach in CDB 1.x, 2.) requirements for evolving the CDB standard to accommodate newer coding schems and 3.) the experimental framework and results from the work in the CDB-X activity.

8.2.1. Current CDB 1.x Attribution Approach

OGC CDB versions 1.0 and higher use a CDB Feature Data Dictionary (FDD) as the controlled vocabulary for attribution. An Excel spreadsheet version of the CDB 1.X Feature Data Dictionary is available on the public CDB wiki [https://external.ogc.org/twiki_public/pub/CDBswg/WebHome/CDB_FDD.xlsx].

The CDB 1.x feature codes and subcodes are primarily based on the DIGEST Feature and Attribute

Coding Catalog 2.1 [https://www.dgiwg.org/DIGEST] (FACC) with additional concepts from a few other sources notably ISO 18025 EDCS [https://standards.sedris.org/18025]. From that website: *FACC) specifies geospatial information concepts used by member nations of the multi-national Defence Geospatial Information Working Group (DGIWG) community. These concepts characterize aspects of real-world entities (or objects) and related properties, including those that are not necessarily visible or have a tangible physical form (e.g., airspace). The DGIWG FACC is a comprehensive dictionary and coding scheme for feature types, feature attributes (properties or characteristics associated with features), and attribute values (domain of feature attributes).*

In the FACC, each major feature category is further divided into subcategories identified by the second character of the five-character code containing an alphabetic value from "A" to "Z". Finally, the third, fourth, and fifth characters of the five-character feature code are a numeric value from 000 to 999. This value provides unique feature type identification within categories. Thus, all features must be identified by all five alphanumeric characters. For example, the feature named "Building" is denoted by the code "AL015". Feature codes are listed in Annex A of the FACC Data Dictionary.

Each feature is associated with a textual description, which provides a human readable dictionary definition for the feature. Each Feature Code is also associated with a short human readable name.

Code	Name	Definition
AA010	Mine	An excavation made in the earth for the purpose of extracting natural deposits. (See also AQ090)
BB150	Landing Place	A place on shore where landing from the sea is possible.
DB180	Volcano	A mountain or hill, often conical, formed around a vent in the earth’s crust through which molten rock, ash, or gases are or have been expelled.
FA001	Administrative Area	An area controlled by administrative authority.
GB075	Taxiway	A prepared surface providing access to/from runways and the aircraft parking area, terminal area, or service area, etc.

An OGC Change Request Proposal [http://ogc.standardstracker.org/show_request.cgi?id=544] (CRP) related to CDB attribution was generated during OGC Testbed 13:

NOTE: An overview of the Testbed 13 activity is provided in the Background Section section of this ER.

8.2.2. Ongoing Standards Development

Originally developed by the Defence Geospatial Information Working Group (DGIWG) in June of 1991 DIGEST FACC was based on earlier standards. DIGEST FACC was most recently updated to the widely used 2.1 version in September of 2000. FACC was officially sunsetted (retired) by DGIWG on October 17, 2012 (see disposition here [http://portal.dgiwg.org/files/7827]) and by the NSG on December 31, 2012 (see disposition here [https://gwg.nga.mil/documents/asfe/DGIWG_FACC.htm]) after a long deprecation period. FACC was replaced by successor standards DGIWG Feature Data Dictionary (DFDD) and associated extension US National System for Geospatial Intelligence (NSG) National Feature Data Dictionary (NFDD). These successor standards were a largely compatible superset of the FACC codes and definitions that did not make any fundamental architecture changes. As a consequence, many commercial and government tools and standards - including CDB 1.0 - continued with FACC 2.1 despite it being officially retired.

The successor standards did bring in additional definitions loosely inspired by ISO/IEC 18025:2005 EDCS [https://standards.sedris.org/18025/] and maritime definitions from IHO ENC S-57 [https://iho.int/uploads/user/pubs/standards/s-57/31Main.pdf]. The NFDD standard eventually was reformulated into the NSG Core Vocabulary (NCV) [https://nsgreg.nga.mil/voc/registers.jsp?register=NCV] and NSG Application Schema (NAS) [https://nsgreg.nga.mil/doc/view?i=5042] which recast the existing standards in terms of logical data models expressed using the Unified Modeling Language (UML) in compliance with the ISO 19109:2015 Geographic Information Rules for Application Schemas [https://www.iso.org/obp/ui/#iso:std:iso:19109:ed-2:v1:en] and the semantic Web Ontology Language (OWL) [https://www.w3.org/OWL] vocabulary. Both the DFDD and NFDD have been retired as well.

The current NAS specification also introduced substantial vocabulary changes describing the metamodel. The most substantial change was that the term "feature" was generalized into the more abstract term "entity" to accommodate vehicles, lifeforms, devices, and abstract ideas like metadata and statistics. The term "enumeration" is still in use to describe a controlled vocabulary of attribute values. However the alternate term "codelist" is also used and the actual values themselves are generally referred to as "listed values." The NAS also introduced a new conceptual model for relating different entities through "associations". Such associations themselves are also entities that mediate complex relationships between entities.

The US Army Geospatial Center codified the NAS 7.0 [https://nsgreg.nga.mil/doc/view?i=3029] reformulation of NFDD into a practical subset targeted at the mission of outdoor environments for the ground warfighter, with encoding standards accommodating limitations of Shapefiles and SQL databases in implementing the general NAS model. This particular application schema became the Ground-Warfighter Geospatial Data Model (GGDM) [https://ggdm.erdc.dren.mil/pages/contents/v3.0.html]. Through a set of various NATO initiatives, the parallel DFDD eventually became the Defence Geospatial Information Model (DGIM) [https://www.dgiwg.org/dgiwg-standards/205]. DGIM also aligned with the NAS 7.0 and considerably overlapped with and took inspiration from GGDM 3.0, albeit with more divergence. The DGIM extended the NAS with some additional definitions from ongoing data model development for the maritime IHO ENC S-100 Universal Hydrographic Data Model [https://iho.int/en/s-100-universal-hydrographic-data-model] and the Aeronautical Information Exchange Model (AIXM) [https://www.aixm.aero] which may be of substantial relevance to CDB for addressing any gaps found in the NAS.

Concurrently, NSG has published an official NAS 8.0 [https://nsgreg.nga.mil/doc/view?i=3031] update. This is the current officially mandated standard with follow-up quarterly updates to address errata, add

extensions, and reformulate documentation. The most recent update at the time of writing is the NAS X-3 [https://nsgreg.nga.mil/doc/view?i=5042] published on June 30, 2020. These updates introduced substantive changes to move away from legacy representations, separated out the traditional FACC 5-letter feature codes and 3-letter attribute codes into a separate "Platform Specific Model" document, separated out view assignments to a separate document, and added definitions for OGC GML concepts. However, the core of NAS today is still recognizably derived from its distant FACC ancestor and still has substantial overlap with it.

A full diagram of the lineage of various geographic information standards is presented here as developed by Geometric Progress LLC for the STE One World Terrain Data Model Working Group.

Data Model Lineage - DIGEST FACC Branch

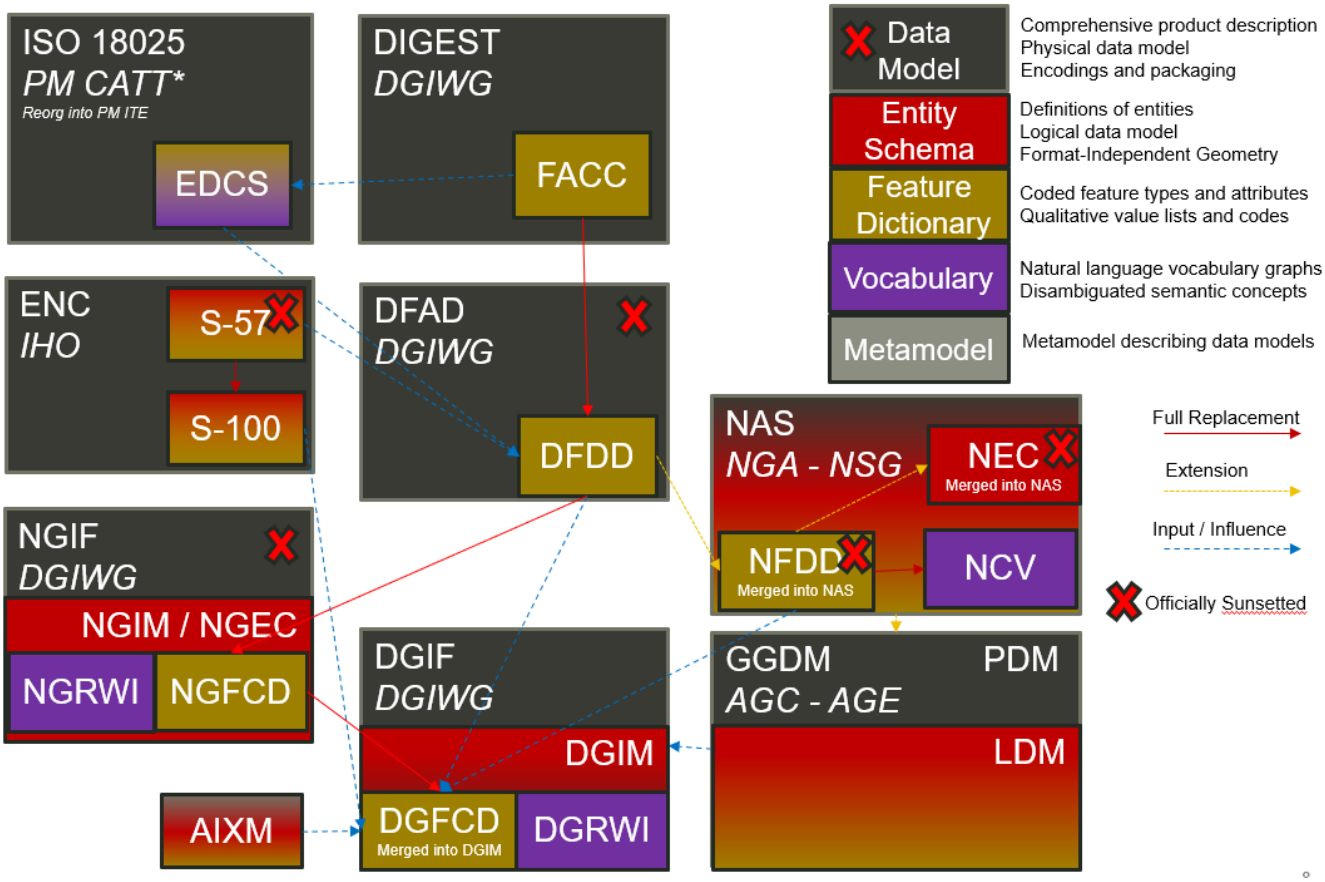


Figure1. Data Model Lineage Chart - DIGEST FACC Branch.

Data Model Lineage - ISO 18025 Branch

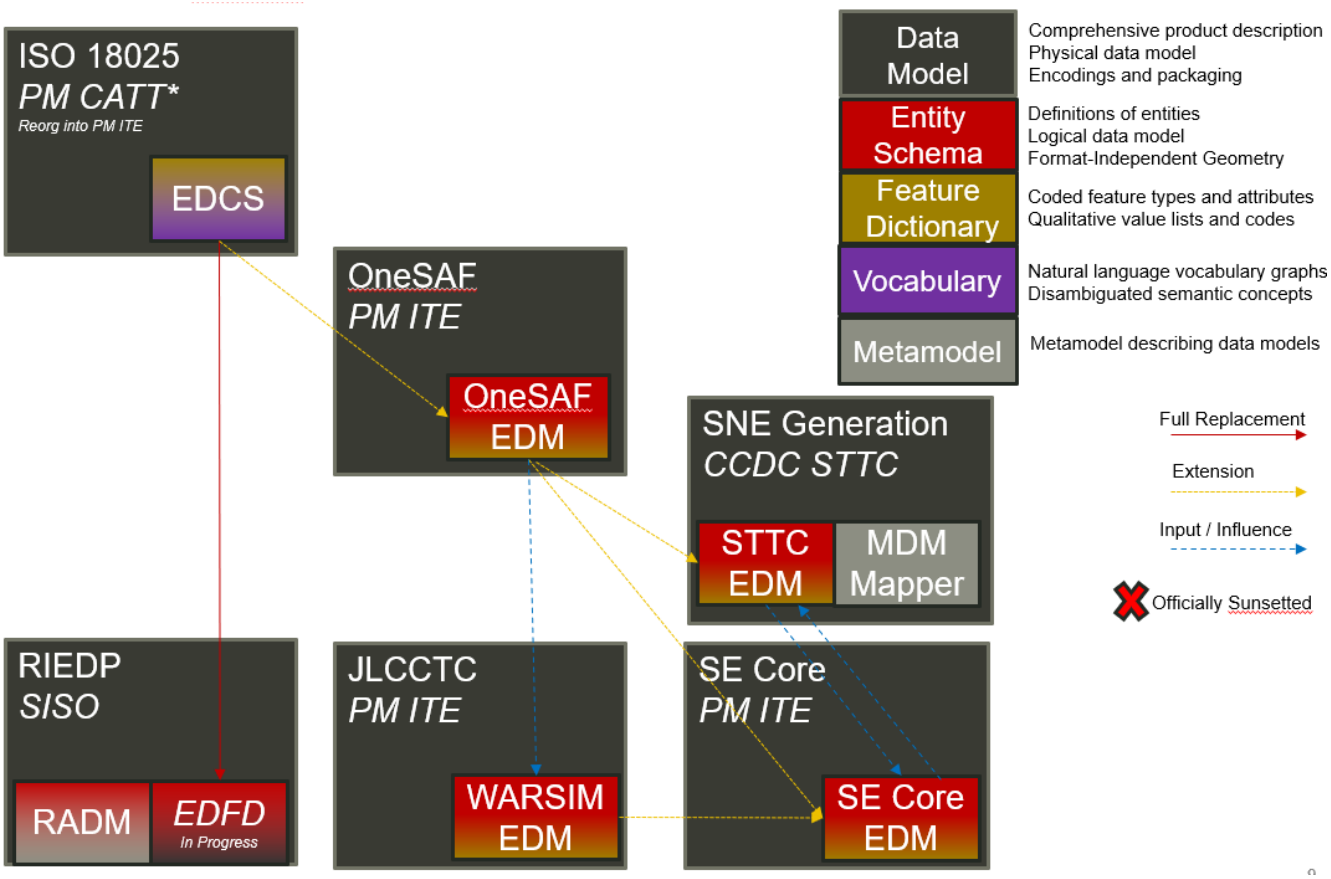


Figure2. Data Model Lineage Chart - ISO 18025 Branch.

Directly relevant to the CDB-X attribution experiment is the adoption of NAS compliance as a design goal and the Ocotber 2019 selection of GGDM 3.0 as the starting point to meet that goal for the US Army Synthetic Training Environment (STE) One World Terrain (OWT) data model. The STE Cross-Function Team saw substantial benefits to moving toward NAS by using GGDM. This was after after verifying that the GGDM met over 90% of current requirements for the existing SE Core use cases [Experiments/Attribution/SE Core to GGDM Gaps.xlsx] for the Army behaviors that were currently met by the existing SE Core Environment Data Model based on ISO 18025 [https://standards.sedris.org/18025]. However substantial gaps for future use cases were identified that were relevant to this experiment. NAS compliance and GGDM interoperability will align the STE simulation and training semantics with existing use cases for geospatial intelligence and operational use. This is directly relevant to the Army's goal of "train as you fight." However, unlike the AGC use case for Shapefiles and GeoPackage SQLite tables for point, linear, and polygon features, the OWT use case primarily intends to deploy GGDM attribution attached to 3D content such as triangle mesh models at the mesh level, the vertex level, the triangle level, and applied to model surfaces via raster coverage texturing. The OWT 3D model approach is also relevant to notional glTF [https://www.khronos.org/gltf/] mesh layers and geotypical and geospecific models explored in other sections of this Engineering Report. There are similar efforts in NGA - notably the FG3D effort - that are also working toward the integration of 3D content with semantic standards.

8.3. CDB-X Discussion and Requirements for CDB-X Experimentation

The following subsections describe the attribution experiments performed as part of the CDB

Research Project.

8.3.1. Experimentation Background - GGDM

An early improvement identified for CDB X was migrating the CDB attribution to NAS compliance using GGDM as the initial set of definitions. This assessment was based on input from current CDB stakeholders and discussions at OGC CDB SWG meetings - particularly given that STE One World Terrain was identified as being a desirable interoperability target. A CDB-X Tech Sprint participant has described work as part of the team recommending a Data Model for the U.S. Army One World Terrain (OWT) effort in multiple presentations to the OGC Interopable Simulation and Gaming Domain Working Group (ISG DWG). The most recent OGC presentation by Greg Peele is titled **"Entities, Attributes, and Enumerants, Oh My! Applying GGDM for Interoperable One World Terrain Semantics"** [https://portal.ogc.org/files/?artifact_id=93666]. While this presentation is out of date relative to OWT development and this experiment the content provides context.

The GGDM is a selection of 624 feature types from the NAS 7.0 (with some duplicates for different geometry representation) to meet the mission requirement of outdoor environment representation for the ground warfighter. There is also a set of associated attribution for each feature type. For attributes with a controlled set of values - also known as enumerants or codelists - the set of values is either explicitly enumerated with integer codes or externally referenced via a separate dictionary of text strings. In principle, every GGDM feature type should match up to an NAS 7.0 logical entity, every GGDM attribute should match up to an NAS 7.0 attribute, and every GGDM enumerant should match up to an NAS 7.0 listed value. All of these should then match vocabulary terms defined in the NCV 2.0. In practice, GGDM did augment the NAS with a small number of additional definitions integrated from the NGA Topographic Data Store (TDS) and the US Environmental Protection Agency Water Resource Database (WRDB).

GGDM adapted the more abstract NAS entity types by binding them to specific geometric representations: Point, Curve (Linear / Line String), Surface (Polygon), and Table (abstract / no geometry). Each geometric binding shared the same 5-letter FACC-like code as specified by the NAS Platform Specific Model (available a separate file in more recent NAS versions) but suffixed the feature name with an appropriate character 'P', 'C', 'S', or 'T'. The attribute definitions were bound to entity types per entity geometry. So in cases where more than one geometry applied to the same entity type, a particular {**Something missing here**} may be present on one but not the other or on both depending on the specification {**Which specification?**}. In the vast majority of cases GGDM only defined one geometry type per entity type. The GGDM developers did clarify that an implementation that did not split out entity types by geometry but used some other mechanism to constrain attribute presence and dataset organization by geometry type to align with GGDM requirements would still be considered compliant with GGDM since in both cases the results comply with the NAS logical data model.

GGDM also organized entity types into a "feature index" that specified broader themes or layers such as Hydrography. These themes were also specific to each geometry type. The feature index also defined five levels of detail: Global, Regional, Local, Specialized, and Composite. Each entity type was cross-referenced to specify in which themes the entity belonged to at each level of detail, or to mark that a feature was not represented at a particular of level of detail. This approach to feature organization substantially diverged from the base NAS standard. The NAS instead defined a set of "view groups" (more abstract) and "views" (more specific) to create a two-level hierarchy of

entity type organization but did not define any levels of detail. The GGDM feature index themes appear to be related to an older version of the NAS "view groups" but the two are currently out of sync. Unlike in GGDM, NAS views are non-exclusive so an entity type may belong to more than one view, although one view is typically marked as the primary view for that entity type. In more recent revisions of the NAS such as NAS X-3, the entity to view assignments are available as a separate document from the main content definitions.

One substantial innovation of NAS and GGDM over earlier standards is moving from a flat table-based feature dictionary to a full logical data model compliant with ISO 19109 which allows for multi-valued attributes, range-valued attributes, complex attributes with named sub-fields, and relationships between features. The NAS logical data model expresses these innovations in pure UML without implementation guidance, while the GGDM defines an encoding to represent them in traditional table-based systems.

For multi-valued attributes, GGDM defines a separate attribute code for each value with the second value and so on suffixing the attribute code with an ordinal index. For example, for FFN "Feature Function", the first value would be FFN, the second value would be FFN2, the third value would be FFN3, and so on. GGDM sets a max cardinality of 3, but there is no technical reason why more values could not be used in other applications.

For range-valued attributes, GGDM splits them into three attribute values of upper, lower, and interval closure. The latter is an enumeration describing whether the lower and upper values are considered part of the range. For example for WDA "Average Water Depth" you would have WDAL "Average Water Depth <lower bound>", WDAU "Average Water Depth <upper bound>", and WDAC "Average Water Depth <interval closure>" as three separate attributes. This is a replacement for the clumsy approach in DIGEST FACC that used enumerations of a predetermined set of ranges for these attributes instead of explicitly specifying the range numerically.

Finally, for complex aggregate attributes and feature relationships, GGDM defines a scheme to take the logical value relationship as a hierarchy and flatten it into prefixed attributes that combine the datatype or feature type with the target attribute code. GGDM defines this flattening such that no attribute code exceeds 10 characters. This means some particularly complex attributes have the prefixed names truncated. An example of a complex aggregate attribute is RIN_RTN "Route Identification <route designation>" - the NAS UML defines Route Identification as a separate class with a Route Designation attribute. An example of a feature relationship expressed as an attribute is ZI006_MEM "Note : Memorandum" which is a reference to a feature of type Note with an attribute value of Memorandum. In some cases the related features are fully embedded in the source feature and thus duplicated for every feature; in others the related feature is referenced by a foreign key using the UFI "Unique Entity Identifer" attribute.

Each of these three cases can also be combined with each other. For example, a multi-valued or range-valued attribute on a related feature, or a chain of multiple aggregated or related features. However doing so tends to quickly hit the 10 character truncation limit for the attribute code. While currently used for Shapefiles and GeoPackage implementations, this particular encoding scheme for complex attributes is not strictly necessary to claim GGDM conformance. Directly representing multi-valued, range-valued, and complex attributes natively by some other mechanism such as JSON, binary markup, or separate SQL tables would still be considered compliant with the GGDM logical data model and the NAS so long as the attribute semantics remains the same. Also, using the label or natural language name for entities, attributes, and enumerations instead of the FACC-like

codes the actual attribute storage would still be considered compliant with GGDM and NAS. This is a physical encoding implementation choice.

Given the historical lineage of the NAS and GGDM, there is a substantial overlap between GGDM and the CDB 1.x Feature Data Dictionary. However, neither standard is a strict superset of the other. NAS and GGDM have changed existing definitions inherited from FACC as well as adding many new definitions. CDB has made substantive changes to add a new concept of "Feature Subcode" that did not exist in prior standards. This was done by bringing in a different set of definitions from ISO 18025, and adding new definitions for aeronautics and building interior components. STE One World Terrain, in particular, had identified gaps in the GGDM standard for building interiors, aeronautics, vegetation, and materials which are all current CDB use cases. Therefore the existing CDB extensions over FACC may end up being complementary to GGDM rather than redundant and may correlate with ongoing standards development in other domains.

8.3.2. Experimentation Goals

Given the strong consensus that adopting NAS using the GGDM as a starting point represents the best path forward for CDB X, an experiment was planned to validate this hypothesis and determine what gaps and difficulties this change would introduce. There was a particular focus on any changes in CDB storage structure would be implied by moving to GGDM. During Phase 3 of the CDB X Tech Sprint, an initial metamodel was created describing the proposed schema for representing the target GGDM data model and a notional SQLite metadata encoding to store it in a more runtime-efficient way than the current CDB 1.x XML metadata.

The first planned experiment was to create a prototype software code library representing the proposed CBD X feature dictionary metamodel. This prototype {**Not sure but I changed future to past assuming the prototype was implemented**} defined runtime classes for each of the metamodel concepts. The prototype also implemented a proof of concept reader that could load both the existing CDB 1.x XML feature and attribute dictionary metadata as well as loading the GGDM 3.0 entity catalog as conventionally formatted in an Excel spreadhseet. The prototype would then finally implement proof of concept support for storing the dictionary content in the proposed SQLite metadata encoding. A stretch goal of also implementing sample XML and JSON encodings for comparison was included. The primary goal of the first experiment was not necessarily to fully implement all of the capabilities, but rather use the prototype to identify and document any deficiences or mismatches in the proposed CDB X feature dictionary metamodel - ideally with proposed corrections - that would interfere with migrating existing CDB 1.x feature data or representing the proposed NAS-compliant dictionary.

Since CDB 1.x and GGDM have essentially compatible semantics of what an entity (feature) type is, the next phase of the experiment was to assess data model mappings between GGDM, TDS, and CDB. This was to determine how cleanly the existing CDB feature types translate to GGDM feature types and identify any substantial gaps in GGDM as well as mappings that lose precision or involve additional complexity. Particularly interesting is identifying how much of the mapping preserves the existing CDB feature code, feature label, or ideally both. There was also a plan to use the gaps identified to suggest a mitigation strategy for filling those gaps either using existing CDB 1.x definitions or from other open standards and to examine similar efforts conducted by SE Core and STE One World Terrain. While initial assessments suggested that attribution and enumerant values would likely map mostly directly due to both CDB and GGDM largely pulling from the same FACC

ancestry, documenting any mismatches we found regarding attribute values was planned. GGDM and NAS entity types and attribution were reviewed for describing feature-level metadata to propose a possible mechanism to implement that in CDB X. Ideally, and as another stretch goal, the subgroup planned to adapt the prototype software library developed for the first experiment to use name and code matching to generate an automated mapping to compare with the manual assessment. However, there was not time to meet that stretch goal.

The third "experiment" is more of a thought experiment to coordinate with the tiling, vector features, and 3D model subgroups to identify what changes to the feature dictionary and data model will imply on changes to structure and content of the CDB datasets - particularly vector and 3D model datasets. This would identify the key areas of standards development for attribution outside of the feature dictionary metadata itself. It may also inspire changes to the CDB dataset structure and content to better align with the target GGDM data model.

8.4. CDB X Experiment Findings

The three experiments conducted successfully generated a number of findings initially captured as Github tickets [https://github.com/sofwerx/cdb2-concept/issues] **{Will need to change this somehow to a publicly accessible Git issue set}**. The prototype code library for the feature attribute metamodel was partially implemented to a sufficient degree to:

1. Identify the mismatches with both the CDB 1.0 feature dictionary and the target GGDM 3.0 data dictionary and
2. Provide partial implementation for both the core SQLite encoding and the stretch goal XML encoding, but not the JSON encoding.

These results generated substantial interest among the STE One World Terrain stakeholders to sponsor the completion of the prototype code library targeted at the One World Terrain data model use case. The subgroup reviewed and assessed a number of mappings and reports between GGDM, TDS, and CDB 1.x to capture the major mismatches and findings. However, the stretch goal of using the code library to generate automated mappings for comparison so that experiment remains as future work. This may be a useful technique for approaching the mapping to NAS X-3 and DGIM. This is proposed as a recommendation. Finally, after substantial discussion on Slack and via Github, the subgroup identified the relatively few areas where changing the GGDM data model had a substantive impact on the CDB dataset storage.

8.4.1. Entity Metamodel Comparison

The CDB 1.x entity metamodel is overall similar but less complex than the GGDM and NAS entity metamodel. This is to be expected since all of these standards derived from the same FACC metamodel. However, the NAS and GGDM have undergone substantial development since then to align with current data model practice. NAS and GGDM renamed the "feature" concept to the more general form "entity" to accomodate phenomena that were not traditionally considered features like devices, lifeforms, vehicles, and abstract ideas. AS such the basic notion of an entity being a particular phenomenon with a unique identity in the simulated or real world that is described by attributes with particular values is still the same. NAS and NCV explicitly define a semantic ontology hierarchy of entity type subclassing that refines very general and abstract entities into specific entity types. This can be very useful for semantic understanding of complex datasets. This

hierarchy is implicit and assumed in GGDM as an application of the NAS rather than explicitly stated. It does not exist at all in CDB 1.x modulo {**What does this mean?**} being implicit for a few items brought in from external standards for building interiors and explicitly via the generalizations (more in terms of aggregation than subclassing) specified by the category/subcategory organization of entity types and by feature subcodes.

Both the CDB 1.x and the GGDM represent entity types using a 5-letter code mostly inherited from FACC, although NAS and GGDM have modified some existing codes and both have added new ones. CDB 1.x specifies entities purely semantically and then specifies a recommended geometry type and data set for each entity type. This as well relies on the semantics of the first two letters of the FACC-like 5 letter code to organize entity types into a two-level category hierarchy for 3D model storage. NAS specifies entities purely semantically. Entities that do have a physical geometry have an associated attribute that may be a point, curve, or surface value or combination thereof. GGDM specifies entities separately per geometry type using a suffix on the entity name and does specify a theme (data set) for each entity type, albeit separately for each of five levels of detail. GGDM and NAS entities may be related to other entities through associations which is a concept that does not currently exist in CDB 1.x but may prove very useful for data de-duplication, feature-level metadata, and annotation. CDB 1.0 additionally defines a specific semantic concept of feature subcode that does not exist in GGDM and NAS.

Primitive attributes are also essentially the same conceptually: They describe a particular quantiative or qualitive property of an entity type using a name, a short code, and a data type, a measurement unit for measured quantities, and constraints on the values. In traditional FACC the attribute codes are always 3 characters. CDB added a number of additional attributes with 4 character codes, many of which are related to model instancing. GGDM attributes are typically 3 character codes for simple attributes. However suffixed attributes for multi-valued and range-valued attribute are 4 characters and prefixed attributes for complex data types and feature relationships may be up to 10 characters. The primitive data types of Boolean true/false, integer, real number, text, and (partially) enumeration are essentially the same in both standards, although the multi-valued, range-valued, and complex attribute value types in GGDM do not have an equivalent in CDB 1.x. While the core concept of attributes is equivalent, the details of constraints, in particular, do vary substantially. Another substantive difference is that attributes are bound to datasets in CDB 1.0 but are bound individually to geometry-specific entity types in GGDM. CDB also has a concept of optional attributes with default values to fill in missing information, whereas all attributes are mandatory in GGDM and no default values exist.

The controlled vocabulary for qualitative attribute values - enumerations, codelists, etc. - is similar conceptually. For closed vocabulary sets, GGDM and CDB 1.x are essentially compatible in that they identify a list of terms and assign them numeric ordinal indices (values may be skipped). For open vocabulary sets that reference external standards, GGDM specifies them using text strings either in a separate sheet in the GGDM Excel spreadsheet or through an externally web-accessible registry. CDB 1.x has no equivalent to this kind of text-based codelist and would currently have to store such values as freeform text with no validation.

Groupings to organize entity types into datasets, collections, and categories are substantially different between CDB 1.x and GGDM and this difference will need to be reconciled.

One concept that exists explicitly in the NAS but is implicit and not stated in GGDM and CDB 1.x is the notion of physical quantities. These describe the types of measurements that may be made for

values. For example, a quantity of "Length" is defined to measure linear distance and the measurement units of "metre" and "foot" are realizations of that quantity. This concept is primarily used to identify which units may be converted to each other and what those conversion factors are.

Based on this comparison, the belief is that the metamodel for CDB 1.x currently is mostly a compatible subset of the current NAS metamodel modulo {**I would different phrasing**} a few mismatches discussed in following sections. The subgroup recommends extending the current CDB XML metadata to add the NAS metamodel capabilities that are currently not supported. The CDB core conceptual model should not mandate any particular data dictionary or content, but rather provide the conceptual and logical metamodel for describing any ISO 19109 compliant application schema to the maximum extent practical. There should be no technical reason why one could not develop an extension profile for CDB for any particular data dictionary that complies with ISO 19109.

With that conceptual metamodel in place, creating a new NAS-compliant CDB Data Dictionary captured in a backward-compatible CDB_Feature_Dictionary.xml and CDB_Attributes.xml (and related files) starting with GGDM is the next step. - Detailed recommendations are provided in follow-up findings {**Need internal linkl**} on how to accomplish this goal. Such an approach would enable a backward compatible path to migrating the current standard to NAS compliance using GGDM (with some modifications) as the encoding mechanism to deal with complex attribution so no structural changes to CDB 1.x vector encoding are needed. However, the subgroup also recommends developing for the CDB X major revision a replacement database metadata strategy that encapsulates the entire data model and data dictionary in a single SQLite file that will be easier for runtime clients to use and query at runtime. This is especially since clients will be expected to have SQLite support anyways if GeoPackage is the vector dataset encoding. CDB X enhancements would also enable developing native representations of complex attributes for newer encodings that may not necessarily need the GGDM encoding simplification approach.

8.4.2. Feature Subcodes not in GGDM

One very concrete difference between CDB 1.x and GGDM is that CDB 1.x defines a built-in concept of "Feature Subcode" in addition to the normal "Feature Code" by specifying the 5-letter code. This feature subcode is stored as a separate attribute in the vector attribution table and is an integer value of up to three digits describing a specific subtype of the broader feature type. The introduction of feature subcodes was a substantial change from the originating FACC standard and no other standards assessed use this concept. OGC CDB ticket 544 [http://ogc.standardstracker.org/show_request.cgi?id=544] highlights that using feature subcodes does not comply with ISO 19109 or the NAS. Relatively few CDB feature types use feature subcodes. However, the ones that do tend to be highly relevant such as power plants, buildings, vegetation, and building interior components.

Based on this assessment, many - but not all of the CDB subcodes - originated from different FACC enumerated attributes playing a more specialized role. For example, the subcodes for AD010 "Power_Plant" are directly traceable to the values of the POS "Power Source" attribute which still exist on the GGDM AD010 "ELECTRIC_POWER_STATION_P" although CDB defines some additional values that are not present in FACC or GGDM such as "Solar", "Wind", and "Internal_Comb". In some cases very general attributes such as FFN "Feature Function", PPO "Physical Product", MFY "Medical Facility Type" and so on are used to make these distinctions in GGDM, particularly in regards to buildings and structures. Due to the lack of definitions for individual vegetation objects and

building interior components in GGDM - as previously identified by STE One World Terrain - the CDB 1.0 feature subcodes for these types are objects are novel and have no counterpart in GGDM.

CDB X cannot include the concept of feature subcode and remain compatible with GGDM, NAS, OWT, or ISO 19109. A mapping strategy needs to be defined and missing semantic definitions will need to be added to the CDB X extension of GGDM. Ideally this would be formulated using the NCV vocabulary so it can be submitted back to the developers of GGDM and NAS for inclusion in future revisions of those standards. The subgroup recommends treating the CDB 1.x feature subcode conceptually as an attribute - purely for mapping purposes - rather than its own concept whose valid enumerated values are different depending on the entity type. Where possible, this attribute should be mapped to existing GGDM or NAS attributes such as POS, FFN, and PPO. In cases where an appropriate attribute exists but not all feature subcodes have valid mappings, the subgroup recommends adding new enumerant values to represent those concepts using the existing CDB 1.0 definitions. In cases where appropriate attributes or entity types do not already exist in GGDM or NAS, additional decisions need to be made. The subgroup believes looking at other standards would be the best first choice. For example the DGIM and its referenced standards IHO S-100 and AIXM may provide substantial definitions for maritime and aeronautics. In the event that no open standard provides suitable definitions, the first decision is whether to create separate entity types for each subcode definition if they are sufficiently different from each other. These can be still be related by subclassing relationships at the logical level as is done in the NAS - or to create a single entity type encompassing all of them and then defining an enumerated attribute to represent the possible feature subcodes. Building interiors merit a separate discussion due to the complexity and role other external standards such as IFC/BIM, CityGML, IMDF, and others play.

This change is possible in a structurally backward-compatible way without changing the schema of the current CDB 1.x XML metadata standard. This approach could be done by simply not using feature subcodes - or more accurately, only ever using feature subcode "000" to avoid breaking the parsing structure. This would be when writing the replacement Feature_Data_Dictionary.xml that captures the NAS-compliant and extended entity types that replace the feature subcodes. Replacement attributes used to map feature subcode would also have to be added to CDB_Attributes.xml file. Once this is in place, feature subcodes could be deprecated, but not removed until CDB X. For CDB X, we recommend simply not including feature subcode as a concept at all and map CDB 1.x databases using feature subcode as a codelist or enumeration attribute.

8.4.3. Mapping between CDB, TDS, and GGDM

The subgroup reviewed sets of existing mappings between NGA TDS 6.0, TDS 7.0, GGDM 3.0, and CBD 1.0 [Experiments/Attribution/GGDM_to_CDB_Crosswalk_20200713.xlsx] that had been developed by Cognitics, Inc. and others to assess the completeness of the mapping from CDB 1.0 to GGDM 3.0. This was primarily focused on entity type and attribute mappings.

The primary findings for entity types is that out of roughly 2,000 CDB entity types (including distinct subcodes) approximately 30% of them have a direct and obvious mapping to GGDM 3.0. Of this 30% that have obvious mappings, almost all of them either match on 5-letter code, on entity name (ignoring case and geometry suffix), or frequently both due to shared lineage from FACC. Some very common entity types did change either code or name between CDB and NAS/GGDM - for example AL015 is Building in CDB but it's AL013 in NAS and GGDM, whereas AD010 is "Power Plant" in CDB and "Electric Power Station" in NAS and GGDM - so it's not accurate to say CDB is a

subset or superset of NAS or GGDM in terms of names, codes, or definitions.

Of the 70% of CDB feature types that did not immediately map to GGDM, the majority are various specific types of buildings and structures that CDB represents as unique entity types or feature subcodes than does NAS or GGDM. In GDDM, many of these concepts are handled as a more generic AL010 Facility, AL013 Building, AH055 Fortified Building, or similarly generic entity type with one or more attribution values specifying the details. The subgroup believes these mappings do at least partially exist - perhaps even for a majority of the entity types - but will require substantial effort to develop and cross-reference to ensure the semantics are compatible and that missing values are added since it is not a straightward name-based match.

There are also substantial gaps in the GGDM data model for particular categories present in CDB 1.x. GGDM lacks any representation for individual vegetation objects (other than EC005 "Tree") and any representation for building interior components except for a handful that also can exist standalone in the outdoor environment. GGDM also lacks any definition of detailed material composition or aeronautically-rigorous light points. CDB 1.x does handle materials and light points as separate conceptual things that are not part of the CDB feature dictionary. Other areas where there are some gaps include climate zones and biomes, detailed aeronautics, detailed maritime features and port structures, and fighting positions and other military-related dynamic terrain obstacles. In retrospect, most of these gaps should be expected because of GGDM's specific mission to apply the NAS to the specific needs of the outdoor environment for the ground warfighter. However the lack of infantry-relevant fighting positions and dynamic terrain obstacles is a little surprising given that mission.

The latest version of the full NAS (currently NAS X-3) provides definitions for many but not all of these gaps so the **recommendation** is to revisit mapping CDB 1.x to the latest full NAS X-3 rather than the GGDM subset to capture the true coverage of the mapping. This approach will also ignore geometry mismatches since the NAS does not separate entity types by geometry. Domains that NAS X-3 does not cover are building interiors and individual vegetation. These will require a separate approach synthesizing findings of multiple existing civilian standards. Maritime definitions not present in the latest NAS may instead be available in DGIM courtesy of IHO S-100. Aeronautics definitions definitions not present in the latest NAS may instead be available in DGIM courtesy of AIXM. The unmet stretch goal of updating software to provide fully automated mappings as a starting point will likely be very useful for a follow up experiment if a suitable NAS and DGIM loader is written.

For attribution, the situation is a bit more straightforward in most cases. CDB has a very limited selection of 66 attributes relative to much larger GGDM and NAS. The migration to NAS will allow for much more detailed semantic representation of the environment with 1,942 different unique attributes at the cost of higher resource use when actually {**Currently?**} used. For attributes with built-in meaning to the CDB structure itself, they {**Who or what?**} primarily did not map to anything in GGDM or NAS. Examples primarily related to model instancing and topology connections such as BBH "Bounding Box Height" as well as the CMIX "Material Identifier" and CNAM "Class Name" foreign keys. This is to be expected and these attributes will likely remain unique to CDB, although the model instance and material attribution may have some synergy with the visual data model being developed for STE OWT.

Due to the shared FACC lineage most of the remaining attributes had straightforward mappings from CDB 1.x to GGDM. However, one unexpected quirk is that many measured attributes in FACC

were integer attributes with very limited (one meter or one degree) precision with alternate attributes specifying real values to full precision. CDB went with the latter for obvious reasons of accuracy. NAS and GGDM amended FACC to change all the integer-valued measured attributes into real-valued attributes to capture the proper intended semantics, and removed all of the alternate forms. Examples include AOO "Angle of Orientation" vs. CDB using AO1 "Angle of Orientation with greater than 1 degree resolution" and WID "Width" vs. CDB using WGP "Width with Greater Than 1 meter Precision". The NAS changes in this regard make sense and simplify the data model to how many vendors were already using it in practice. While it does not affect CDB mapping, similar changes were made in CDB to remove the alternate FACC attribute forms that specify measurements in non-SI units. The NAS instead provides a set of quantity and measurement definitions to allow implementations to store measurements in any unit if desired while specifying the standard unit for each attribute. The main exception is for the few cases such as SPD "Speed Limit (MPH)" vs. SPM "Speed Limit (KPH)" where the distinction between the units is legally or semantically relevant and not just a measurement detail.

Summarizing the methodology proposed for mapping CDB to a NAS-compliant data model:

1. Start with existing CDB 1.0 to GGDM mapping to identify core set of compatibility for outdoor environment (revisit attribute based mappings for subcodes).
2. Conduct full mapping from CDB 1.0 to NAS X-3 (or later) to capture NAS compliant mappings and document NAS gaps.
3. Conduct mapping to DGIM followed by IHO S-100 for maritime gaps in mappings to NAS, develop NCV definitions for these gaps.
4. Conduct mapping to DGIM followed by AIXM for aeronautics gaps in mappings to NAS, develop NCV definitions for these gaps.
5. Coordinate with OGC CityGML, STE OWT, and other stakeholders to synthesize a building interior data model and develop NCV definitions for these gaps.
6. Coordinate with STE OWT and civilian environmental agencies to synthesize a detailed vegetation data model and develop NCV definitions for these gaps.
7. Any remaining gaps will require new semantic development to create NCV vocabulary and frame in NAS compliant logical model.

The subgroup recommends capturing the new NAS-compliant definitions into a new CDB_Feature_Dictionary.xml. Further the subgroup recommends also defining a ruleset format in XML or some other easy to use encoding - possibly leveraging technologies like ShapeChange - for defining the translation rules from CDB 1.0 feature dictionary to the NAS-compliant feature dictionary in a reproducible way. While most translation rules will be a simple 1:1 mapping from feature type to feature type, some will rely on conditional mappings of values for feature subcode or other attribute values and a few are fairly specific. This particular approach could also be used to develop other sets of standardized data model mappings from or to ISO 18025, vanilla DIGEST FACC 2.1, or different versions of NAS or DGIM to improve the migration path of interoperabillity between data sets.

8.4.4. Existing FDD Metadata Missing Attribute Details

The existing CDB 1.x Standards specify more details in the feature and attribute dictionaries as

human-readable descriptions and graphics than are actually present in the machine-readable XML. These details are relevant in migrating to the NAS-compliant attribution model and it is unreasonable to expect clients to hard-code them based on reading the Standard. The two key items that are missing at the machine-readable level for the current Standard are the definitions of which attributes should be present on each dataset and the list of valid enumerant values for each attribute. Both of these gaps can be filled in a straightforward backward-compatible way by adding XML elements to existing metadata as a minor revision in CDB 1.3 or beyond.

Linking attributes to datasets could be organized either by specifying a list of datasets under each attribute in the in the CDB_Attributes.xml, or by specifying a list of attributes under each dataset in the Datasets.xml. In either case, this linkage would also be modified by an elements pecifying whether the attribute's presence is "Preferred", "Supported", or "Deprecated" to match the existing CDB 1.x human-readable specification. Additionally the subgroup proposes adding status for "Mandatory", "Optional", and "Retired" status as discussed later regarding default values and mandatory attribution.

For defining enumerants, the subgroup recommends adding an <Enumeration> element under the <Value> element in the CDB_Attributes.xml and for each valid enumerant value, add a <Term> element that specifies the name/label, the integer code, and the full definition/description. The set of valid enumerants for each attribute is already defined in the CDB 1.x specification, it just needs to be captured in this XML.

8.5. Feature Relationships

One innovation of the NAS above the older NFDD and FACC specifications is the adoption of a mechanism for specifying entity relatioships to model complex phenomena. This can be used in a wide variety of ways:

1. To specify spatial-topological relationships such as connectivity, containment, adjacency, and attachment;
2. To specify aggregation of feature types into broader containers or general categories;
3. To specify descriptions of detailed properties shared between multiple features;
4. To specify chains of provenance and derivation; and much more.

Strictly speaking in a UML sense the generalization relationship representing the link between child entity subclass and parent entity superclass is also such a relationship, although this is usually modeled specially rather than as a general relationship.

Despite the broad range of possible applications, relationships really only fall into one of two structural metamodel categories: Direct and mediated. Direct relationships describe very simple relationships where the only relevant information is the source entity, the target entity, and the type of relationship. The types of direct relationshiops might include:

- Association - Completely generic relationship (UML)
- Generalization - Subclass to superclass (UML)
- Realization - Implementation to interface (UML)

- Aggregation - Object to collection (UML)
- Composition - Part to whole (UML)
- Containment - Object to container
- Connection - Physical attachment link between siblings
- Adjacency - Physical proximity without attachment between siblings
- Description - Object to detailed information about it
- Representation - Logical object to physical geometry or encoding

Mediated relationships are necessary when the relationship itself needs further description such as attribution. In UML and ISO 19109 application schemas, relationships by themselves cannot have attributes. The resolution is to introduce mediating entity types that represents the relationship itself, and then define a direct relationship from the source to the mediating entity and from the mediating entity to the target.

The NAS tends to represent direct relationships as complex attributes with implied relationship semantics. Strictly speaking, this could be derived from the exact UML connector though. GGDM then deploys these direct relationships by flattening them out into prefixed attributes where the attributes of the target feature are embedded on the source feature. The NAS also defines a small number of mediated relationships, but these are not currently deployed in GGDM.

The subgroup proposes adding relationship support to CDB 1.3 in a backward-compatible way by creating a new Relationships.xml that describes the set of direct relationships and places constraint on which entity types can relate to which other entity types. This would be by relationship type and also what is the maximum cardinality of that relationship. The source, target, and if relevant mediator entity types will be references into the CDB_Feature_Dictionary.xml. A sample XML file for this might hypothetically look like:

```
<Relationships>
 <Types>
  <Relationship><Name>Description</Name><Definition>blah de blah de blah
</Definition></Relationship>
  <Relationship><Name>Connection</Name><Definition>blah de blah de blah
</Definition></Relationship>
 </Types>
 <Constraints>
  <Relationship>
   <Source>AP030</Source>
   <Target>ZI016</Target>
   <Type>Description</Type>
   <Cardinality>1</Cardinality>
  </Relationship>
  <Relationship>
   <Source>UG000</Source>
   <Mediator>UK004</Meidator>
   <Target>AL013</Target>
   <Type>Description</Type>
   <Cardinality>Unbounded</Cardinality>
  </Relationship>
 </Constraints>
</Relationships>
```

Actually implementing feature relationships should be delegated to the vector, raster coverage, and 3D model encodings. However, this subgroup believes that the defined constraints would enable the use of foreign key columns (i.e. by integer ID referencing into a different table) to represent relationships in a GeoPackage encoding with each modeled entity including mediating entity stored in its own relevant table by dataset.

8.5.1. Grouping Features into Datasets and Categories

One sustantial metamodel difference is how entity (feature) types are organized into datasets and categories. CDB 1.0 currently provides two different ways of organizing entity types that are used in unrelated ways.

The first, relying on the two-letter prefixes of the FACC-like codes specified in the Feature_Data_Dictionary.xml metadata file, organizes entity types into categories and subcategories. For items derived from FACC, this organization generally is semantically coherent in which similar or related entity types end up in the same category. The same is generally less true for entity types in CDB that were not derived from FACC; such extensions, particularly the entity types starting with 'U' and 'V', tend to be more organized by origin of definition than by semantics although the subcategory usually still provides some semantic grouping. The category/subcategory grouping is used primarily to decide folder paths and file names for 3D models.

The second, specifying separately in the Datasets.xml metadata file, organizes entity types into separate datasets (layers) which then in CDB 1.0 imply different files and file formats for each dataset. The datasets represent both raster coverages such as Elevation as well as 3D model data sets and vector datasets; the vector data sets can then be further implemented as point, linear, and

polygonal sublayers. This concept of grouping is core to the current CDB 1.0 standard and dictates filenames, formats, and content. As a substantial divergence from GGDM, the CDB 1.0 standard specifies the list of valid attributes at the dataset level rather than for each entity. CDB X may relax the storage organization impact due to experiments with GeoPackage containerization, but the groupings may still affect internal tables in the GeoPackage.

NAS and GGDM do not exactly have equivalent concepts to either one of the CDB 1.0 grouping types. The closest concept in NAS is the concept of "view groups" and "views" which are a two-level hierarchy of organization of entity types. Entity types may belong to more than one view, and each view may only belong to one view group. The two-letter FACC-like prefixes in NAS do not have any normative meaning since they are defined by a separate "Platform Specific Model" rather than being a core part of the entity type definition, although in practice new entity types in NAS still select 5-letter codes in a way that mostly maintains semantic groupings based on the first two letters. The closest concept in GGDM is the feature index, which is recognizably similar to the NAS view groups but not consistent with them, but are geometry specific.

The original design from Phase 3 only accounted for NAS view groups and views as a replacement for category and subcategory. This was done by assigning each entity type only to its primary view. However the experimentation showed this was insufficient to model CDB 1.0 due to the core role datasets play in the CDB storage layout and attribution. The fact that attributes are specified at the dataset level is a curve ball. In NAS only entities may conceptually have attributes - not containers or groupings - which made the initial design insufficient to migrate CDB 1.x databases into the new logical model.

Going back to the most abstract level - the NCV - gave some insight on how to reconcile this mismatch. At that level, every definition is simply a vocabulary term, with containers being just a different type of term that can have children. This is not exactly what we needed, but the participants realized that combining that with a concept of relationships from the NAS would enable us to generalize containers such as datasets to also be entities.

So the specific recommendations to map CDB 1.x to the NAS conceptual model are as follows:

1. Define entity types that represent each dataset and associate the relevant attributes to each dataset as described in the prior section.
2. Add a "generalization" relationship from each dataset to NAS entity type ZI031 "Dataset".
3. Define entity types for each category and subcategory.
4. Add an "aggregation" relationship from entities to datasets that contain them.
5. Add an "aggregation" relatioship from entities to their containing subcategory.
6. Add an "aggregation" relationship from subcategory to its containing category.

In CDB 1.x all of these constructs would be implicitly set up by the definition of the Datasets.xml and Feature_Data_Dictionary.xml, whereas in CDB X these could be explicitly represented in the data model SQLite storage and logical model.

In CDB X, we recommend the NAS-compliant replacement to do the following:

1. Define entity types that represent each dataset, aligning with NAS/NCV where possible.

2. Define attributes for each entity type as specified by NAS and GGDM.
3. Add a "generalization" relationship from each dataset to NAS entity type ZI031 "Dataset".
4. Use the existing NAS view groups and views as the replacement of category and subcategory, defined as entity types.
5. Add an "aggregation" relationship from entities to datasets that contain them.

As part of a backward-compatible change to CDB 1.x, the subgroup recommends adding an <Aggregation> element sequence to the current <Subcode> element to enable specifying additional containers for entity types beyond the category, subcategory, and dataset containers implied by the current structure. This will provide a migration path to generally specifying of arbitrary depth of views and containers compliant with ISO 19109. The subgroup also recommends adding a <Generalization> element to each <Subcode> element to capture the parent/child subclassing relationship for entity types defined by NAS.

8.5.2. Per-Entity vs. Per-Dataset Attributes

As explained in the previous section about feature groupings, one divergence of CDB vs. NAS and GGDM is that CDB 1.x defines which attributes are valid at the dataset level, whereas NAS and GGDM define the set of valid attributes and their associated constraints specifically for each entity type (and in GGDM, unique for each geometry type). To migrate toward NAS compliance, CDB X will need to specify the set of valid attributes per entity type. The previous sections proposes a recommendation of how to adapt the existing per-dataset attribute definitions to the proposed CDB X conceptual model to maintain backward compatibility.

However, the inverse can also be done: Extend the CDB_Feature_Dictionary.xml to add a new XML <Attributes> element under the existing <Feature_Type> element to list the set of valid attributes for that particular entity type referencing the definitions present in CDB_Attributes.xml. Each such element could include an optional XML modifier for each attribute to specify that the attribute only applies to a particular geometry type to represent GGDM geometry-specific constraints. Another XML modifier could apply the same "Mandatory", "Preferred", "Optional", or "Deprecated" status as currently prescribed for linking attributes to datasets. This <Attribute> element could also specify the same set of constraints such as range, precision, etc. to override the global definition for that specific feature type, although in practice most will probably just use the global definition.

This recommendation would enable implementing NAS-compliant per-entity attribution constraints within the current CDB 1.x structural framework via backward-compatible extensions while allowing the prior per-dataset definitions to remain in place as a deprecated element while clients migrate.Ths approach could then be fully retired in CDB X as a breaking change. The global attribute definitions are still useful from an NCV deep semantics standpoint of capturing which attributes across different entity types have the same semantic meaning.

8.5.3. Multi-Valued Attributes

GGDM and NAS added the conceptual model for multi-valued attributes and many existing FACC attributes were retrofitted to become multiply-valued due to many entity types logically having more than one material, function, product, etc. Notably, many of the attributes that are likely to be mapping targets for CDB feature subcodes become multiply valued such as FFN "Feature Function",

POS "Power Source", and PPO "Physical Product". For NAS compliance, the CDB X logical data model must support multi-valued attributes.

The GGDM provides a very convenient way to represent multiply-valued attributes that preserves backward compatibility with clients that do not understand multi-valued attribution. The attribute is defined with its name and code as normal, and the corresponding Shapefile or SQL column provides the first value of the multiple values. Each additional value is suffixed with an ordinal and defines the next value. So, for example, FFN column defines the first "Feature Function" value, FFN2 column defines the second "Feature Function" value, and so on. GGDM imposes a hard limit of 3 values, but no such limit needs to exist in CDB. Semantically, GGDM also imposes two useful constraints: Values must be ordered by significance with the most significant (i.e. highest relevance) value first, and must also be filled out in order: FFN2 cannot be set if FFN is blank, for example. Clients not aware of multi-valued attributes will simply read the first and most relevant value and see the remaining value columns as simply unknown attributes, whereas clients aware of multi-valued attributes can interpret them directly.

This change can be accomplished with a backward-compatible update to CDB 1.3 to update the <Attribute> element in CDB_Attributes.xml to have an optional <Cardinality> element with <Min> and <Max> children specifying the minimum and maximum number of values permitted for the attribute. Clients not aware of this change can simply ignore this element and treat the attribute as a scalar value reading the most significant value. If the <Cardinality> element is not present, a default value of 1 for both <Min> and <Max> may be assumed. Actual storage of attribute values in the Shapefile encoding will follow the GGDM ordinal-suffix column convention described previously.

For CDB X, the cardinality of attributes must be properly represented as well. More options exist in terms of value encoding. For example, having a single column encoded using "Well-Known Binary" list of values would be an option, as well as sticking with the GGDM ordinal-suffix convention. For 3D model encodings such as glTF, multi-valued attributes are already inherently supported in binary buffers due to the use of visual vertex attriutes of position, normal vector, etc. For advanced simulation, particularly for civilian infrastructure, extending the NAS concept to allow weighted multi-valued attributes may also be useful. This could be where the data can specify the percentage or a weight relevance factor of each value to precisely state what fraction it contributes to the overall entity. This would an area of new research if adopted.

8.5.4. Range-Valued Attributes

GGDM and NAS added the conceptual model for range-valued numeric attributes specified as a lower bound, an upper bound, and a closure enumerant specifying whether the left, right, or both endpoints of the range are included. This is primarily used for representing attributes on larger-scale aggregate features such as EC015 "Forest" where the value represents an average or range of uncertain values over a whole curve or surface rather than a precise value at a specific point. FACC had previously represented such attributes instead using enumerations that provided a set of predetermined range buckets to choose; that approach was very imprecise particularly at the highest and lowest range buckets. For NAS compliance, the CDB X logical data model must support range-valued attributes. These types of attributes can particularly be useful for procedural random generation of point models and other detailed content from much coarser aggregate linear and polygon features.

The GGDM provides a mechanism for representing range-valued attributes: The base attribute code is suffixed with 'L' for the lower bound, with 'U' for the upper bound, and with 'C' for the closure enumerant to define three Shapefile or SQL attribute columns that fully specify the range. This approach works, but is not backward compatible with clients that do not understand range-valued attributes which is a drawback for an incremental change. Therefore the subgroup proposes a slightly different approach for backward compatibility to add this to CDB 1.3: Specify the mean or midpoint of the range as the base attribute code without a suffix, specify the width of the interval as the attribute code with the 'W' suffix, and the endpoint closure enumerant with the 'C' suffix as in GGDM. Clients not upgraded to work with range-valued attribution would then simply read the midpoint of the range as a single scalar value which is a reasonable fallback. Attributes could be marked as range-valued by adding an additional <Mode> element as a child of the <Range> element in <Attribute><Value> with possible values of "Scalar" and "Interval" with "Scalar" being the assumed default if missing; additional modes could be added in the future if applicable. The actual vector or model encoding could still decide to only use the single scalar value or the full range value depending on data content availability. While GGDM range-valued attributes are always real numbers, there is no inherent reason to disallow integer attributes from being range-valued as well to simplify the XML schema.

For CDB X, little would change except range-valued attributes would be natively part of the data model. The proposed mean-width-closure attribute encoding could still be used, or alternate encodings packing the values into triplets in a binary structure. An area of future research may be adding metadata to specify additional details of the range such as the random distribution in use such as "Uniform" vs. "Normal" vs. "Exponential" - such an extension would be extremely useful for procedural generation of individual feature attribution from containing zones or aggregations.

8.5.5. Text Patterns and Codelists

The CDB Standard already provides for text attributes and also for specifying a minimum length constraint. For basic text attributes, the current CDB approach is sufficient. However, GGDM and NAS define two more advanced variants of text attributes that would benefit from a backward-compatible update to specify them. The first is the notion of "Structured Text" which is a text attribute that matches a particular layout and pattern. Examples of structured text includes dates, Basic Encyclopedia Number, and URIs. This could easily be accommodated in CDB 1.3 by adding to the <Attribute><Value> tag in the CDB_Attributes.xml a new tag for <Pattern> which is simply a regular expression that the values must match.

NAS and GGDM also define a concept of codelist attributes, which are effectively open-ended enumerations stored as text values rather than as integer codes. In many cases, these codelists are references to external ISO standards like country codes. Much like precisely defining the existing integer-based enumerations, we recommend explicitly capturing the codelists into the <Attribute><Value> element as the child element <Codelist> specified by one of two mechanisms: either explicitly with a sequence of <Term> elements specifying the value and definition, or implicitly for ISO standards and such by providing the name and URI to the referenced standard.

This change does not imply any change to vector encoding and would be carried forward exactly as described into CDB X. Clients not aware of the pattern constraint or the codelist definition would simply treat such attributes as freeform text as is currently done.

8.5.6. Instance, Class, and Extended Attributes

CDB 1.x provides three different encoding techniques for representing attribution. Instance-level attribution is the most traditional approach. This is the case where each attribute is treated as a column and each entity is treated as a row, so every entity has a unique value. Class-level attribution provides a level of indirection in which a separate table has each unique row of attribution and entities reference the row providing their attribution via the CNAM foreign key attribute. This approach allowed for de-deduplication if multiple features had exactly the same attribution for a subset of their attributes. Extended attributes were a third representation using a separate table in which each attribute value was a separate row with a column for feature ID, attribute code, and attribute value. Currently, attributes in the CDB_Attributes.xml identify whether they are preferred, supported, or deprecated for each of the instance, class, and extended encodings.

No similar concept exists in NAS or GGDM, and teh subgroup believes this concept should not be present in CDB either, at least not at the feature data dictionary level. This sort of thing instead represents a physical encoding choice that may vary substantially between different databases based on actual content and capabilities of the container formats. In particular, all breakout groups had a consensus that Extended Attributes were no longer necessary as they were not widely used, were extremely inefficient, and are totally unnecessary with a GeoPackage or glTF encoding. The subgroup saw some potential applicability for class-level attribution as a compression technique at the encoding level, possibly using foreign key constraints in GeoPackage. Feature-level metadata and information feature relationships - described separately - provide a more structured way for providing common descriptions shared by many features that better solve the use case when semantically many features do in fact share the same values for any reason other than chance.

The subgroup recommend fully deprecating Extended Attribution in CDB 1.3 and to be removed completely in CDB X. The subgroup also recommends deprecating in CDB 1.3 in CDB_Attributes.xml the <Level> tag identifying which attributes are preferred, supported, or deprecated for each attribution encoding and to consider all attributes to be instance-level attributes at the logical level. Clients already must be prepared to handle all attributes at both the instance and class levels, so no change will occur in that regard. Whether class-level attribution should exist as an optimization technique for vector and model containers remains an open question that should be resolved at that level without any feature data dictionary implications.

8.5.7. Mandatory vs. Optional Attributes and Default Values

GGDM considers all attributes mandatory and thus no default values (aside from the global defaults of false, 0, and the empty string) are defined. Many but not all enumerations have a value for "unknown." For the operational purposes for which GGDM was designed, this is sensible. However, for the much wider modeling and simulation use case for which CDB is designed, many attributes may not have data availability, some may truly be optional. Further, in any case modeling and simulation will need reasonable default values to assume for procedural generation of content to support behaviors and other modeling goals.

The CDB Standard already provides mechanisms to specify default values for attributes specifically for each dataset that uses them in the Defaults.xml file. The exact elements and formatting for this is awkward and relies on specific text formatting of the <Name> element, but it sufficient to meet the requirement. However, currently no mechanism exists to specify the default value for an

attribute when it used for a particular entity - the case in which it is mostly semantically meaningful - and there also no mechanism to specify whether a particular attribute is required or optional for a particular dataset or entity.

The subgroup proposes adding these definitions as a backwards-compatible extension to CDB 1.3. The subgroup proposes:

1. Adding the <Default_Value> element to be a valid child of each <Attribute> defined under a <Feature_type> in CDB_Feature_Attributes.xml (specified previously).
2. To define with the same <Default_Value> syntax as currently present in the Defaults.xml, minus the <Dataset> element.
3. To specify the default value of an attribute for a particular feature regardless of what dataset it is in.

As discussed in the prior section about capturing existing constraints into the XML metadata, the subgroup also recommends capturing the <Status> of an attribute relative to both its containing dataset or it containing feature where these bindings are specified as one of the following statuses:

- Mandatory - The attribute must be present on this dataset or feature and it is validation error for it not to be set.
- Preferred - The attribute is strongly recommended to be present on this dataset or feature.
- Supported - The attributed may be present on this dataset or feature with no judgment on whether it should be present.
- Optional - The attribute may be present on this dataset or feature but is considered purely optional and may be safely omitted.
- Deprecated - The attribute may currently be present on this dataset or feature but is dicouraged and may be removed in the future
- Retired - The attribute was allowed in a previous version of the standard to be present on this dataset or feature but is currently no longer allowed (not strictly necessary but may be more useful for error messages vs. attributes that are just completely made up).

For range-valued elements, additional elements would need to be added to support defining the default value's range width and closure type in a way that clients would ignore if they do not understand it. If a default value should be mulit-valued, that case should also be defined in a way that clients unaware of it can ignore it.

Clients unaware of the new feature attributes defaults would simply ignore them and not use them, as well as ignoring the <Status> element added in various places.

For the CDB X, all of the feature defaults and status can be directly consolidated and represented in the CDB X SQLite data dictionary without issue.

8.5.8. Organizing Attributes by Domain

One prior observation is that the migration to NAS will extend CDB from 66 attributes currently to a possibility of 1,942 NAS attributes. This could be even more if augmented by additional standards for maritime, aeronautics, and building interiors. While great from a completeness standpoint, this

has a corresponding impact on resource use and many CDB clients will only be interested in a small subset of those attributes for a particular use case.

The subgroup proposes developing a new conceptual mechanism for organizing entities and attributes into one or more functional domains such as "Physics", "Visual", and so on so that a particular user can quickly identify which entities and attributes might possibly relevant to them and only load and use those attributes. Tools could then easily use this to tailor full repository CDBs into slimmed-down CDBs intended for edge computing or other streamlined use cases. Domains could be composed into larger domains to make it easy to describe any particular use case.

Aside from tailoring to optimize a CDB for a particular use case, domains could also be useful to clients to provide default functional behavior for broad classes of entity types without explicit simulation support for each entity type. For example, every entity type in a "Solid" domain could have a default interaction to block line-of-sight and weapon fire simulations in a client device unless specifically overriden by entity type or attribution, whereas entities not in that domain would not affect such simulations by default.

As a new concept in the metamodel, substantial care would need to be taken to careful define the semantics of domains. One research question would be whether domains are a type of entity and thus can arleady be described using the entity and aggregation mechanisms proposed in prior sections, or whether this should be done as its own separate orthogonal concept.

8.5.9. Metadata vs. Attribution

Metadata and provenance are a major discussion topic in many geographic information standards activities. Historically, metadata tended to mean one of two orthogonal things: Provenenance metadata describing the authorship, acquisition, accuracy/precision/uncertainty, and creation process of particular data, and indexing metadata which provides simplified summaries of the data to make it easier to discover through search techniques. {**This sentence is quite long and convoluted. Any way to break it up?**} provided for by the CDB organizational structure itself, but the former is a major gap of substantial importance.

The reality is that, as the NAS development proves, the historical distinction between metadata and attribution is largely artificial and unnecessary. Structurally, metadata is attribution and attribution is metadata. The only difference is which entity types provide for which attributes and how they are related. For provenance metadata, typically many features share the same provenance metadata since they were generated from the same data sources using the same authorship and workflows - the provenance itself can be represented as a single entity with attributes and other reference entities such as workflow steps, with all features with the same provenance referencing the same provenance entity.

The NAS and GGDM provide for the following entity metadata types, for example:

- ZI001 Source - representing a single data source
- ZI002 Restriction Information - description of a unique set of commercial and classification restrictions controlling data
- ZI004 Data Process Step - describes an individual workflow step of an overall production process

- ZI005 Geographic Name Information - representing a particular abstract named thing
- ZI006 Note - a general sticky note that can be attached to just about anything
- ZI026 Physical Object Metadata - general information about cartographic usability of an object with associated process workflow
- ZI031 Dataset
- ZI039 Entity Collection Metadata
- ZR102 Acquisition Process Metadata
- ZR106 Data Quality

In a purely flat Shapefile-like encoding, GGDM binds these metadata entities to individual features by taking all of their attributes and applying them as prefixed attributes to the source entity, duplicating the metadata values for every entity that uses them. This approach is simple and can be done today in CDB 1.x, although it is very resource inefficient. With the CDB X GeoPackage encoding, these metadata entities can instead be stored in their own dataset tables inside of the GeoPackage and simply referenced by the relevant features using a foreign key attribute implementing an entity-level relationship. It would even be possible to use a table inside of the GeoPackage to provide the collection-level data for all contained tiles, datasets, the GeoPackage as a whole, or even the containing CDB as a whole.

Subtantial metadata-specific work would need to be invested to determine the exact details of which NAS metadata is necessary for the CDB NAS profile and the exact representation of metadata in the target encodings. The subgroup proposes using the latest version of the NAS cross-walked against the NSG Metadata Foundation (NMF) and the ISO 19115 Geographic Information Metadata standards.

8.5.10. CDB Vector Geometry Data Model vs. Other OGC Standards

One minor divergence the subgroup found in OGC CDB 1.x vs. the other OGC standards was the definition of vector geometry. OGC Simple Features provides an implementation of ISO 19125 that describes the vector feature model used in essentially all modern geographic applications with core geometry types Point, Curve (typically realized as LineString) and Surface (typically realized as Polygon). The existing CDB 1.x standard is inconsistent about the terminology used, and in the machine-readable CDB_Feature_Attributes.xml the non-standard terms Lineal and Areal are used instead of Curve and Surface (as used in GGDM) or LineString and Polygon (as used in OGC Simple Features for the equivalent geometry). The set of valid geometry types is also not defined in the CDB 1.x XML metadata aside from implicitly in the CDB_Feature_Attributes.xml as references in the <Recommended_Dataset_Component> element.

These are very minor issues, but should be resolved. The subgroup proposes updating the CDB 1.3 specification to consistently use the same terms from OGC Simple Features - whether it is "Curve" and "Surface" or "LineString" and "Polygon" is a matter for debate - to also align those terms with the equivalent terms in the latest NAS which were themselves pulled in from OGC GML. The subgroup also proposes, for completeness, creating a new Geometries.xml metadata file for CDB 1.3 that captures the list of valid geometry types in that particular CDB. While most clients likely will have specialized handling of geometries, this can be used as a validation check for a client to make sure it can process all geometries specified in the metadata file. Unfortunately, to maintain

backward compatibility, the geometry names as used in CDB file name structure and CDB_Feature_Attributes.xml <Recommended_Dataset_Component> values cannot be fixed. That change will have to wait until CDB X to fully align the specification with other OGC standards.

Strictly speaking, the latest NAS defines geometry as a type of entity in the same sense as a feature, lifeform, or idea, including giving each geometry a 5-letter FACC-like entity code; the vertex geometry of a feature is then treated as a particular attribute for entities that have physical geometry at the logical level. However, it is unclear whether this view of geometry would be practically useful in any way; by nature specific encodings like GeoPackage and glTF treat geometry specially.

One final question to investigate later on geometry is whether there is any benefit to expanding CDB to support the full geometry capabilities specified by OGC GML. The increased generality would have a tradeoff with the performance constraints for which CDB was designed.

8.5.11. Entity Dictionary Storage Design

As part of the code prototype experiment, the subgroup started experimenting with a SQLite table design for representing the entirety of the proposed CDB X logical data model and code to read and write it. This implementation was not completed but based on the partial efforts found no reason to believe there was an substantial technical barrier to doing so. The subgroup also observed that by laying out the metamodel in UML, it would be possible to create equivalent XML and JSON schemas - perhaps through tools like ShapeChange - to offer multiple possible encodings of the CDB X metadata.

Based on cross-referencing the GGDM, NAS, and NCV with CDB-specific requirements, the following tables were notionally identified as important. This structure is intended to be experimental rather than prescriptive - we believe this should be an area of further research for CDB X. It may also be appropriate to conduct experiments on using an SQLite data dictionary vs. XML or JSON data dictionaries to see if the impact on clients is noticeable.

- Entities - core vocabulary of entity types
  - ID - primary key
  - Label - NCV label
  - Code - FACC code
  - Definition - full definition
  - Generalization - base Entity type
  - Type - Entity, Geometry, Category, Dataset
- Quantities - core vocabulary of measurement physical quantity types
  - ID - primary key
  - Label - NAS label
  - Definition - full definition
- Measurements - core vocabulary of specific measurement units
  - ID - primary key

    - Label - NAS label
    - Quantity - foreign key referencing which quantity this measurement measures
    - Prefix - prefix when used (typically empty, example could be $ for currency)
    - Suffix - suffix when used (i.e. m for metre, may be empty if suffixes not used)
    - Multiplier - multiplication factor to go from this unit to base quantity unit
    - Divisor - divisor to divide by to go from this unit to base quantity unit (keep exact precision when this is how conversion is defined)
    - Offset - offset to addto go from this unit to base quantity unit (typically 0 except for temperature)
    - Derivation - mathematical expression to describe how derived measurement is composed of products of powers of base units
- Datatypes - core definitions of attribute value types and base constraints, de-duplicate attribute value definitions
    - ID - primary key
    - Label - name of datatype
    - Primitive - underlying primitive Boolean, Integer, Real, Text
    - Minimum - minimum value (integer and real variants)
    - Maximum - maximum value (integer and real variants)
    - Precision - preferred decimal precision for real attributes
    - Closure - whether minimum and/or maximum are included in valid range
    - Mode - Scalar (0) or Interval (1)
    - Measurement - the default measurement unit if applicable (also used for min/max)
    - Cardinality - maximum cardinality of values (1 is scalar, more than 1 allows multiple values)
    - Length - maximum length of text values
    - Pattern - regular expression pattern that text value must match
    - Enumerated - if true, then controlled vocabulary (enumeration values if integer or codelists if text) applies from Codelists table
- Attribute - core definition of attribute vocabulary (independently entity types)
    - ID - primary key
    - Label - NCV label
    - Code - FACC code
    - Definition - full definition
    - Datatype - the default datatype for this attribute (optional)
    - Quantity - the physical quantity this attribute measures, if applicable
- Codelists - definitions of enumerated value vocabulary
    - ID - primary key

    - Datatype - foreign key to parent datatype
    - Filter - foreign key to EntityAttribute if this codelist should only apply to a particular entity and attribute binding
    - Index - integer value if applicable (enumerations)
    - Code - text value if applicable (codelists)
    - Definition - full definition
- EntityRepresentations - table to bind entities to particular geometry representations
    - ID - primary key of binding
    - Entity - foreign key to entity type
    - Geometry - foreign key to geometry entity type
    - Container - foreign key to category or dataset entity type
    - Label - override for entity label when used in this context
    - Code - override for entity code when used in this context
- EntityAttributes - table to bind attributes to particular entities with datatypes
    - ID - primary key of binding
    - Entity - foreign key to entity-representation definition the attribute applies to
    - Attribute - foreign key to attribute definition
    - Datatype - foreign key to datatype specifying attribute value type and constraints
    - Label - override for attribute label when used in this context
    - Code - override for attribute code when used in this context
- Relationships - core set of relationship type definitions
    - ID - primary key
    - Label - name of relationship type
    - Directed - boolean true if directed, false if bidirectional
- EntityRelationships - table to constraint how entities can relate to each other
    - ID - primary key of binding
    - Source - foreign key to source entity type
    - Mediator - foreign key to mediator entity type, if applicable
    - Target - foreign key to target entity type
    - Cardinality - maximum number of target entities that a source feature may relate to
    - Definition - detailed definition of the purpose of the relationship

8.5.12. Data Dictionary Versioning, Changes, and Extensions

One challenge with maintaining long-running data repositories is that data model standards evolve over time. The vast distance between FACC 2.1 and NAS X-3 is a great illustration of that. For CDB X, the subgroup believes that capturing the exact version of data model used and, ideally, the lineage

and authority of each definition will be useful to clients for managing their data over periods of years and decades.

The subgroup identified the following concepts relevant to data dictionary configuration management (which also overlap with metadata standards):

- Agent - an organization, standards body, or individual that performed or approved some action.
- Standard - representation of a complete specific version of a data dictionary definition.
  - Author - the authority that originally created a particular definition.
  - Publisher - the authority that approved a particular definition for inclusion in a particular release version.
  - Title - the name and concept of a particular compilation of definitions - e.g. GGDM, NAS, CDB, OWT.
  - Version - the specific iteration of a particular titled compilation of definitions, typically a version number.
  - Creation Date - the date on which a particular version of a titled compilation was authored.
  - Publication Date - the date on which the particular version of a titled compilation was published.

The available lineage of any particular feature dictionary - including the current CDB 1.x Feature Dictionary - would consist of a set of Standards referencing a set of Agents that authored and published them. Individual definitions would then be tracked through a sequence of Actions describing what was done with the definition, each Action referencing the Standard in which it occurred:

- Action - a particular change related to a definition.
  - Author - the agent that created that particular action (may be different from overall standard author).
  - Publisher - the agent that approved that particular action for inclusion (may be different from overall standard publisher).
  - Creation Date - the date on which the action was authored.
  - Effective Date - the date on which the action should actually take effect.
  - Standard - in which standard this particular action was taken.
  - Action Type - the specific action that was taken.
    - Creation - a new definition was created from scratch without referencing any existing Standard.
    - Adoption - a definition was integrated from another Standard, which must be referenced - subsequent adoptions will override prior actions.
    - Amendment - an existing definition was modified relative to its most recent creation or adoption.
    - Application - an existing definition was bound to a new use case e.g. an attribute was added to an entity or a listed value was added to an enumeration.

- Deprecation - a definition was marked for deprecation to begin the process to sunset its use in the standard.
- Retirement - a definition was retired and is no longer valid for use in the standard.

This experiment was purely a thought experiment of how to manage multiple versions of data dictionaries over the lifetime of a single CDB. Among other things, this would enable different containers in the CDB to reference different versions of the standard to enable data model updates without invalidating older data, or to migrate a particular CDB between different data dictionary standard versions. Whether any of this is a desireable use case is a separate question for the CDB SWG to answer. The one use case where tracking the lineage of a particular definition is of utmost important is the integration of non-NAS definitions. these need to be clearly distinguished so that the appropriate work can be done to propose them for NAS adoption or filter them out for use cases that mandate NAS-only attributes. While it was not the intent of this thought experiment, some of these concepts may also be useful for CDB content revisioning.

8.5.13. Relationship to Materials

The current CDB 1.x standard specifies a simple yet fairly robust physical material system that is more advanced than what is present in NAS and GGDM. Detailed material information is critical for modeling and simulation, particularly for sensor modeling, weapon fire penetration modeling, blast effects modeling, and chemical/biological/radiological/nuclear/explosive (CBRNE) modeling.

The GDGM and NAS approach provides simple enumeration attributes describing material composition of particular feature types - examples include MCC "Structural Material Composition", VCM "Vertical Construction Material", PYM "Pylon Material Type", and BMC "Bottom Material Type". The enumeration values for these material types tends to be limited and fairly broad - examples include "Wood", "Metal", "Concrete", and "Brick". Most of these attributes are multi-valued, although no mechanism exists to describe the proportion of each material or assign them to individual components. Thus the best detail you can typically obtain is to say an entire building is made of brick or an entire river has a bottom of soil. While this baseline capability is important for simulations that don’t need more detail, it is insufficient for physically-accurate simulations. Regrettably, the enumerations aren’t even consistent between different attributes: "Wood" might have value 5 for one attribute and value 15 for a different attribute, and may or may not have the same actual definition. This is a flaw in the NCV vocabulary for materials due to literally importing the FACC 2.1 material enumerations that NSG is aware of.

The CDB 1.x approach defines a similar set of base materials that serve as primitives in the material system. These base materials are roughly similar to the NAS material enumerants, but more detailed and independent of any particular attribute or enumeration. There is one such base material table in the CDB metadata. More complex materials can then be defined at the tile and model level using composite materials that combine base materials using a primary substrate, an optional surface substrate, and any number of subsurface substrates each with a specified thickness. These materials can then be applied to raster material coverages or to assign materials to model components. However, this approach is defined completely independent of the feature and attribute data model.

The gap of GGDM and NAS regarding materials was identified as a problem by the STE One World Terrain Data Model Working Group. Based on those findings, US Army CCDC is funding research

efforts to create a NAS-compliant data model for materials. The baseline goal for the material data model is to fully represent CDB 1.x and SE Core materials including substrate definitions. The model also goes substantially further in defining an entity model for base materials that includes physics parameters affecting kinematics and electromagnetism phenomena, weighted chemical and isotope composition, and radiation and other emitters. This research will also explore other ways of combining materials such as conglomeration. The research goal of this effort is to define a material system that not only can provide a semantically rich material model but also provides enough specific quantitative detail that each simulation can interpret each material the same way in terms of inputs to physics effects. This goal will be demonstrated with a material library implementing the actual parameters (or at least decent guesses) for the coarse material enumerations currently defined in the NAS as well as showing detailed materials applying to mesh surfaces and volumes through texturing.

The subgroup recommend leaving the CDB material system as-is for CDB 1.3, but coordinate with STE One World Terrain and other stakeholders to integrate the material data model and material library concepts into a NAS-compliant data model for CDB X.

8.5.14. Relationship to Lights

The current CDB 1.x Lights.xml and Lights_Client.xml defines a data model of sorts for light name hierarchy and light rendering properties. While the subgroup did not conduct a detailed investigation or mapping, the superficial assessment suggests that these definitions can be refined into a NAS-compliant data model and incorporated into the main feature data dictionary by modeling them as entity types, attributes, and entity-level relationships. Some of the relevant concepts may be present in the latest NAS, and the AIXM data model may go into more detail. This is an item that should probably be left unmodified in CDB 1.3 and investigated further for CDB X.

8.5.15. Relationship to Model Components

While a detailed investigation of model components was not conducted, a brief glance at Model_Components.xml and Moving_Model_Codes.xml suggests that it would be extremely straightforward to reformulate these in terms of the proposed NAS-compliant data model as entity types, attributes, and enumerations with entity-level relationship constraints. This is an item that should probably be left unmodified in CDB 1.3 and investigated further for CDB X. If this content is to be included in CDB X, that would be a good time to do this reformulation taking advantage of the fact that most - possibly all - of these definitions are likely to be present in the current NAS version.

8.5.16. Building Interior Considerations

Building interiors - and underground structures, having similar considerations - introduce substantial complexity in the data model and attribution. CDB 1.x currently specifies a building interior data model that is mostly a literal encoding of the US Army OneSAF Ultra-High Resolution Building (UHRB) data model. Each building component item - structural items such as walls, organizational items such as floor levels and rooms, and fixtures and furniture - are represented as an entity (feature) type with a CDB-invented 5-letter FACC-like code typically starting with 'U'. Unfortunately, the UHRB specification is no longer publicly available, effectively orphaning the CDB 1.x definitions. Even more unfortunately, building interiors is also a near-complete gap in the NAS and GGDM entity definitions, so this leaves CDB X with substantial work to do.

Building interiors are a specialized use case - many aeronautics, maritime, and large-scale ground environment simulations simply do not need that level of detail. However, emerging applications in architecture, engineering, and construction (AEC), autonomous navigation for robotics, serious games for training, and high-fidelity live/virtual/constructive simulations modeling urban warfare critically rely on rich building interior data models. There are a number of different existing building data models out there for various purposes: the Industry Foundation Class (IFC) is used as a Building Information Management (BIM) interchange format in CAD and AEC, OGC CityGML and associated standards are actively being developed for multiple communities to represent from individual building components all the way up to full cities, and Apple has recently submitted a new Indoor Mapping Data Format to OGC as a community standard primarily aimed at autonomous navigation.

This has also been identified by the STE One World Terrain Working Group as a critical gap in the OWT data model, with an initial interim proposal submitted based on cross-referencing the existing CDB 1.x definitions with other open standards. As part of the OWT assessment, Greg Peele conducted a rough initial cross-walk between different building interior standards [Experiments/Attribution/Interior Assessment and Correlation.xlsx] and took freeform notes on the key elements found in IFC 4.3 [Experiments/Attribution/Assessment of IFC 4.docx]. One deeply problematic finding on this cross-walk is that there are substantial incompatibilities and gaps between these different standards. Another substantial problem is that while IFC is the most widely used for AEC and CAD interchange, it is particularly complex yet does not sufficiently constrain its semantics for true interoperability: Many attributes describing building interior components are unconstrained freeform text in the natural language of the author rather than a controlled vocabulary. If an implementer is lucky, that freeform text might or might not reference municipial, provincial, or national standards where the building is constructed. If not, then all bets are off. There is no off-the-shelf solution that will meet the CDB X or OWT needs right now. Compared to other standards, Apple IMDF is much more limited in scope but relatively precise - it is unlikely to add anything new semantically but can act as a good data input source.

Given CDB's relationship to OGC, the proper step forward for CDB is to cross-reference the existing CDB 1.x definitions to CityGML, IndoorGML, and similar efforts - ideally keeping the same 5-letter FACC-like codes but using CityGML labels where possible. One substantial question to resolve is whether to use existing NAS and GGDM definitions for elements that could be either indoor or outdoor, or to define separate indoor entity types and use the NAS entity types only for freestanding outdoor items. Examples of such dual-use entity types would include AL175 "Courtyard", AL195 "Ramp", AL260 "Wall", AQ150 "Stair", and AP040 "Gate". Some artifacts introduced by the literal UHRB mapping can be cleaned up in the current CDB 1.x feature dictionary: in particular, a number of duplicate feature type definitions exist in the current CDB 1.x feature dictionary with different 5-letter codes, and a number of abstract feature types that aren't actually useful were mapped in from UHRB as well.

CityGML and IndoorGML can also provide a set of useful definitions for room types, fixtures and furniture, and other items to a richer level of detail while preserving interoperability between OGC standards. Items that are in CDB but are missing from CityGML should also be cross-walked to IFC if possible, acknowledging the difficulty of that task: Extensive ongoing efforts have attempted to reconcile the BIM/IFC approach with CityGML for quite some time. The subgroup also recommends ongoing coordination with Army Geospatial Center, NGA, and STE One World Terrain as multiple agencies also look at this problem. In any case, representing the full breadth of interior content will

require the development of substantial amounts of vocabulary and attribution.

8.5.17. Impacts of Attribution Changes on Vector Encoding

Summarizing all of our findings into the impact on the CDB vector encoding: Surprisingly, there really isn't a substantial impact, . Migrating CDB to NAS and GGDM will of course change the attribute column names in Shapefiles and in the proposed GeoPackage encoding, and will change the possible F_CODE values and in some cases attribute data types. However, this does not necessarily imply any particular changes to the vector encoding file names or content storage beyond that. If (optionally) the dataset names are aligned with NAS views, then of course those names would change. Everything that we have proposed can still be done in the existing Shapefile encoding in CDB 1.3, although the GeoPackage encoding will make some of it easier to model directly and much more storage-efficient (particularly foreign keys for feature-level relationships). We identified ways that multi-valued, range-valued, and codelist attributes can be implemented such that existing clients unaware of them would simply see the first and most significant value, the range mean value, and freeform text values instead, which is a graceful degradation. The vast majority of the changes apply to the CDB XML metadata and associated implications on the logical data model and its proposed future replacement in the CDB X SQLite data model.

8.5.18. Impacts of Attribution Changes on 3D Models

Summarizing all of the group's findings into the impact on the CDB 3D model encoding: The impact is relatively minor. The main change is that the current CDB 1.x naming scheme models would result in different filenames for models representing entity types whose name or 5-letter FACC code changed between CDB 1.x and NAS/GGDM. Adopting the newer glTF encoding or other advances may open up additional opportunities for more detailed attribution on 3D models, but there is no fundamental reason why the proposed attribution changes to align with NAS in CDB 1.3 could not work with the existing OpenFlight models.

The relationship between 3D models and light definitions and material composition if those are reformulated into a NAS-compliant data model would be on substantive breaking change if adopted for CDB X. The subgroup does not propose making any changes to these definitions for CDB 1.3 and careful consideration should be given on how to best approach that problem.

8.6. Summary of Recommendations

- Adopt NAS-compliant logical entity-attribute model for CDB X with extensions for CDB use cases
  - Store all aspsects of feature and attribute dictionary in single SQLite file for portability and runtime performance
  - Use GGDM 3.0 as the initial starting point for definitions to populate CDB X data dictionary
  - Match up remaining CDB non-GGDM feature types and subcodes with latest NAS definitions where possible, matching subcodes to attributes where relevant and adding missing enumerant values where necessary (with associated NCV vocabulary)
  - Augment NAS definitions with other open standards and new development
    - Match up missing maritime definitions to DGIM followed by IHO S-100 where possible, define NCV vocabulary for such integrated definitions

        - Match up missing aeronautics definitions to DGIM followed by AIXM where possible, define NCV vocabulary for such integrated definitions
        - Coordinate with OGC, NGA, and STE OWT to develop replacement building interior data model incorporating IFC/BIM, CityGML, IMDF, and other open standards
        - Coordinate with OGC, NGA, STE OWT, and civilian environment agencies to develop detailed data model for vegetation
        - Create data model and vocabulary for material and light point definitions and capture into existing material and lightpoint libraries
        - Define NCV vocabulary and NAS-compatible entity and attribute types for CDB feature types and subcodes totally missing in all other standards
    - Remove CDB feature subcodes entirely; migrate to existing and new feature and attribute types instead in NAS-compliant structure
    - Define entity types for CDB datasets and define "aggregation" relationships from feature types to containing datasets
    - Capture feature-level, dataset-level, and database metadata as NAS-compliant attribution meeting the NSG Metadata Foundation (NMF) and ISO 19115
    - Define functional role domains and create mechanism to organize attribution by domain for tailoring to runtime devices
- Delegate entity and attribute physical encoding choices to vector and 3D model containers instead of specifying globally
    - Deprecate extended attributes entirely, to be removed in CDB X
    - Delegate containerization of entity types (one per table, multiple in same table specified by F_CODE attribute, etc.) to vector and model containers
    - Delegate decision whether to use class or instance attributes to individual vector and model containers rather than global data dictionary
    - Delegate decision of whether to use FACC-like codes, NAS labels, or natural language names for entity types, attributes, and values to vector and model containers
    - Delegate decision of whether to flatten complex feature relationships and attributes used GGDM attribute prefixing to vector and model containers
    - Delegate decision of whether to flatten multi-valued and range-valued attributes using GGDM attribute prefixing to vector and model containers
    - Specify minimum and maximum cardinality of multi-valued attributes in feature and attribute dictionary, allow containers to use a lower maximum if using GGDM attribute prefixing encoding
- Define backward-compatible extensions in CDB 1.3 to add constructs necessary to move toward NAS-compliant attribution
    - Capture proposed NAS-complaint replacement feature dictionary in existing CDB metadata XML with necessary extensions
    - Only use feature subcode 000 in replacement dictionary and deprecate use of feature subcodes to be removed in CDB X

- Add mechanism to mark numeric attributes as interval ranges (existing non-upgraded clients should see still attribute as single-valued and read mean value from associated unsuffixed attribute, use suffixed attributes for deviation and closure for upgraded clients to read)
- Add minumum and maximum cardinality elements for attribute definitions to specify mininum and maximum element count for multi-valued attributes (existing non-upgraded clients should just see attribute as scalar base value and will only read the first value from associated content, will see ordinal-suffixed attributes as separate attributes)
- Add list of valid attributes to datasets in CDB 1.x metadata XML files to match existing human-readable specification
- Add list of valid enumerants for each attribute in CDB 1.x CDB_Attributes.xml file to match existing human-readable specification
- Add list of valid attributes for each entity type as extension to CDB 1.x Feature_Data_Dictionary.xml to implement NAS-compliant per-entity attributes
- Update CDB 1.x CDB_Attributes.xml to allow specifying text pattern constraints through <Pattern> element and text codelists for text attributes via <Codelist> element
- Update CDB 1.x Feature_Data_Dictionary.xml for each feature to specify its generalization (base) entity type via <Generalization> element
- Update CDB 1.x Feature_Data_Dictionary.xml to add <Aggregation> element to define additional associated category for an entity type, or parent category for a category
- Existing category and subcategory XML structure will add implicit definitions and aggeregation links for the category/subcategory items as used by CDB 1.0 for model storage

8.6.1. Phase 3, Day 3

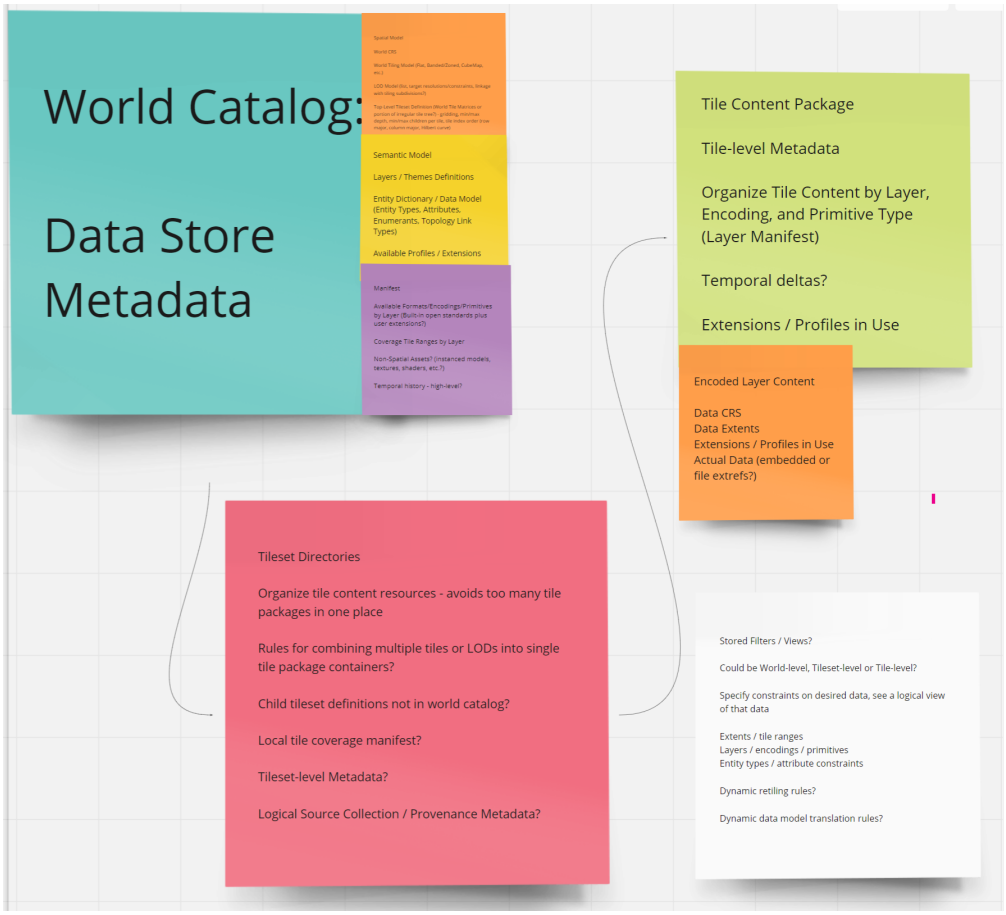


Figure 2. Greg Peele's Whiteboard from Phase 3 Day 3.

8.6.2. Phase 3, Day 4

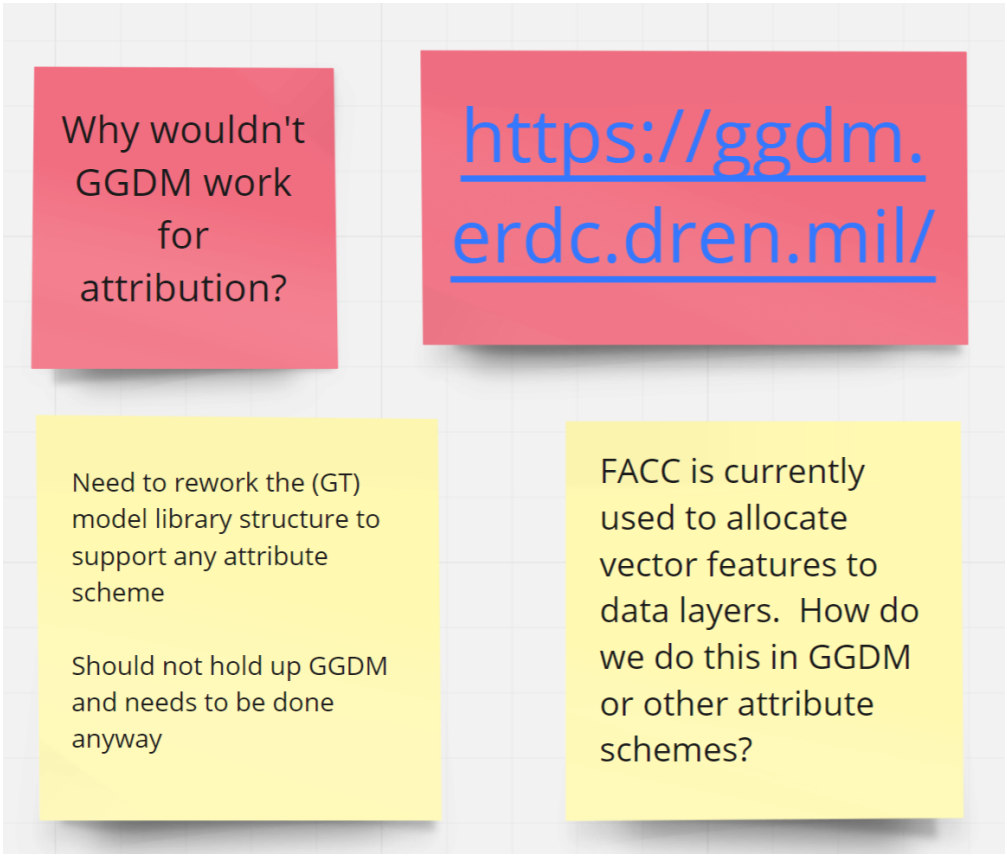


Figure 3. 'Attribution will be in GGDM, Prove me Wrong' Day 4 Sign.

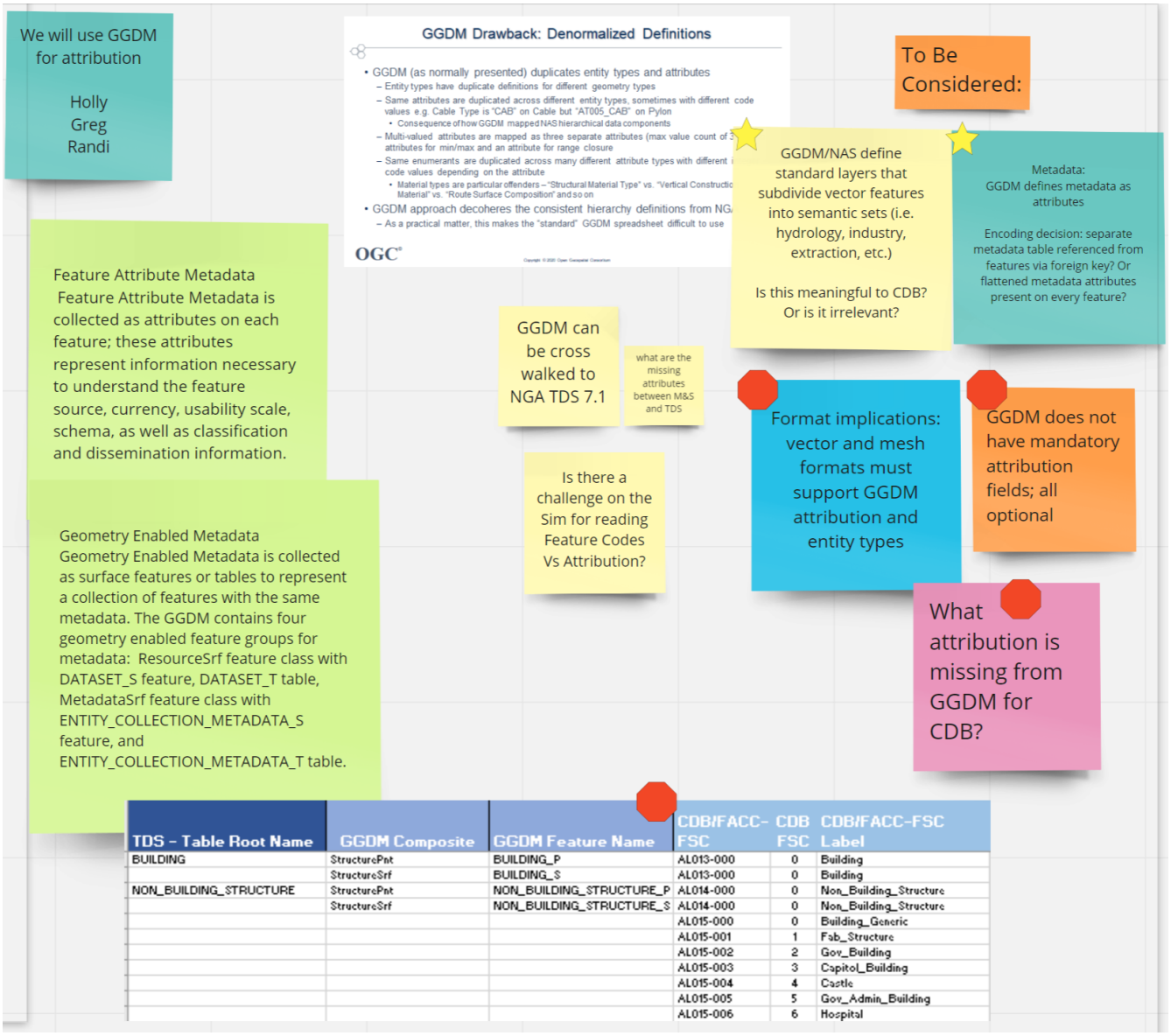


Figure 4. Attribution Day 4 Whiteboard.

8.6.3. Phase 3, Day 5

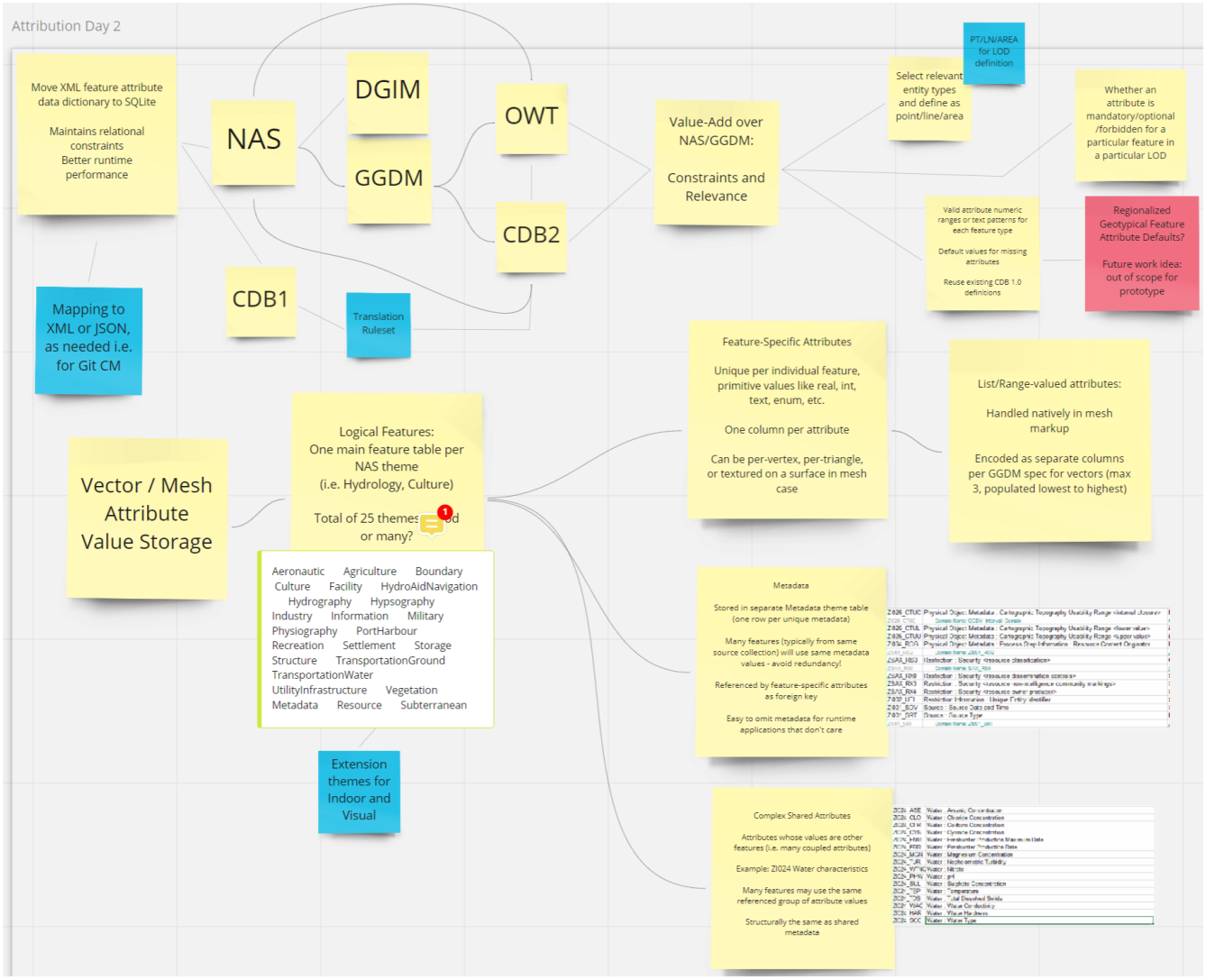


Figure 5. Attribution Day 5 Work in Progress Whiteboard One.

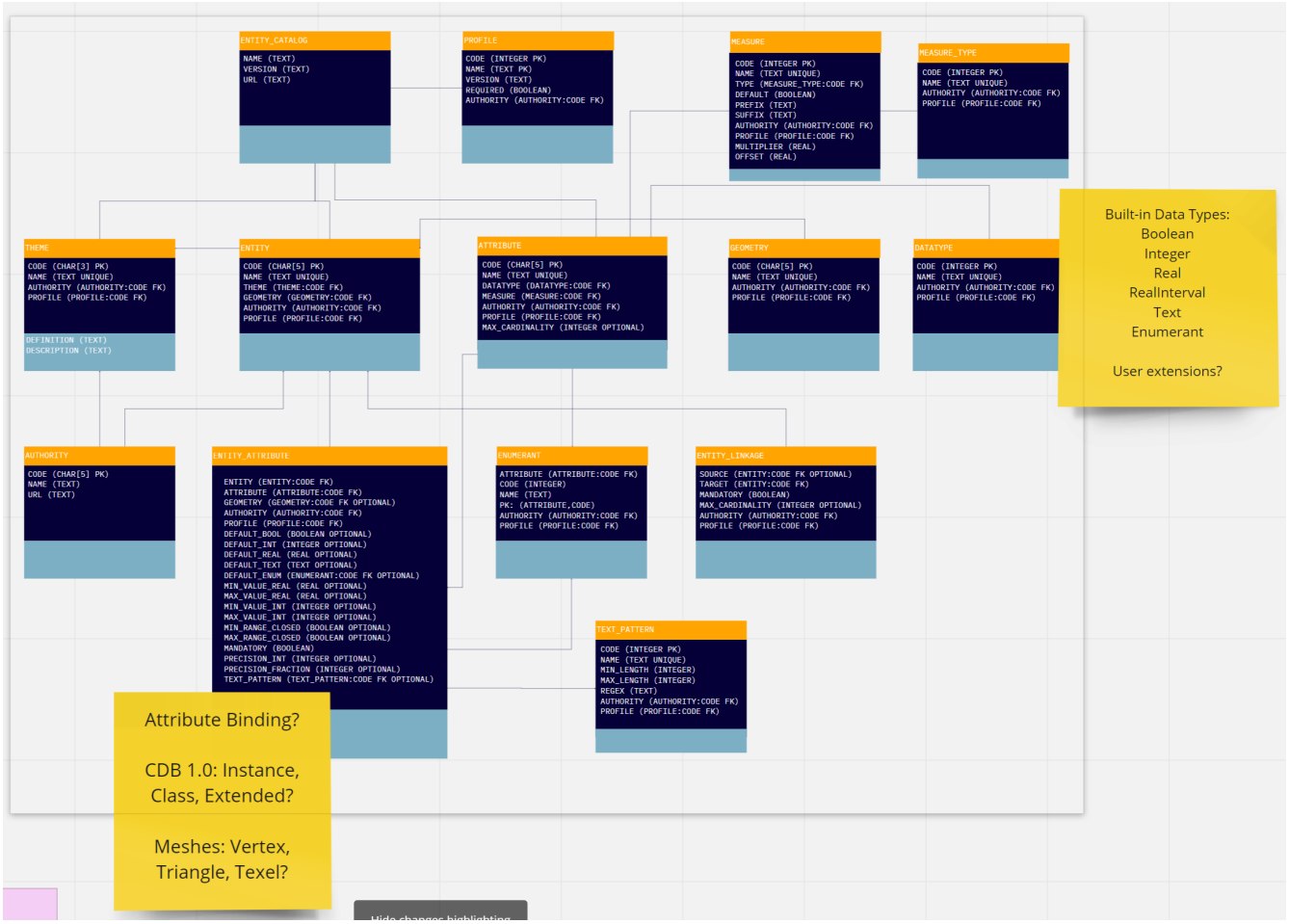


Figure 6. Attribution Day 5 Work in Progress Whiteboard Two.

Chapter 9. CDB X

9.1. Performance of CDB Vector Data in Large GeoPackage Containers

9.2. Abstract

To test potential approaches to storing CDB vector data in GeoPackage, the Vector Data in GeoPackage Subgroup, hereafter referred to as the Subgroup, performed several different experiments to determine the viability of storing very large volumes of vector data in a smaller number of large GeoPackage files rather than a very large number of small Shapefiles. Some of the experiments also set out to determine if there are upper limits in how much CDB content can be stored in a GeoPackage. This document details the Subgroup's approach and results of the experiments.

The experiments demonstrated that GeoPackage/SQLite is capable of meeting the performance and size constraints for vector data in a CDB if properly defined {**CDB or GeoPackage?**}. The experiments also demonstrated that from a performance standpoint doing a direct conversion of existing Shapefile content into GeoPackage(s) may not be feasible. To gain the most from using GeoPackage as a container for large vector datasets, some approaches to level of detail (LOD) and other CDB content should be reconsidered. This includes the way significant size is used in CDB.

The Subgroup also found that the design the schema is very important, including proper index design, when using GeoPackage. Shapefiles are very simple for developers to use since there is not a lot of flexibility when it comes to how to store data. Conversely, GeoPackage is very flexible. As such, how a GeoPackage is structured and indexed can have a major impact on performance. If the exact schema and indexes for GeoPackage are properly defined as part of the CDB-X, users can expect very good performance, even for large datasets.

NOTE	The OGC CDB Standards Working Group will use the work performed by the participants in this Sprint but also further augmented by the OGC Testbed 16 GeoPackage experiments and findings related to processing, storing, and querying extremely large vector datasets.

9.3. Objective

The purpose of these experiments was to examine performance in searching for and interacting with CDB features stored in very large unitary GeoPackage containers. The design intent of the large datasets consisted of three main considerations:

- Be large in terms of feature counts.
- Be CDB-like in terms of feature attribution.
- Be 'representative' of real-world feature density over a large geographic area.

The source data was prepared for the USA, including Alaska, Hawaii and Puerto Rico. Testing,

however, was limited to only the Continental US (CONUS).

9.4. Data Sources

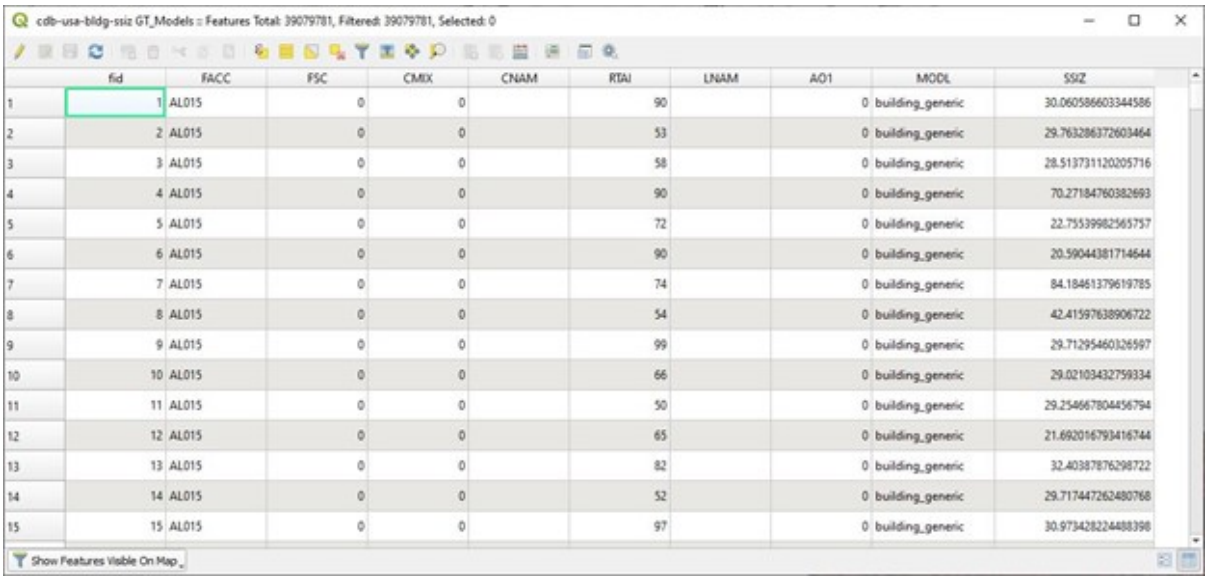
TO achieve the experiment objectives, four GeoPackage containers were populated:

1. cdb-usa-bldg-ssiz.gpkg
2. osm-hydro-to-CDB.gpkg
3. osm-roads-to-CDB.gpkg
4. combined-features-conus.gpkg

These are now described.

9.4.1. Data Source 1 - cdb-usa-bldg-ssiz.gpkg:

This GeoPackage contains 39,079,781 point features and requires 5,249,290,240 bytes of storage. The following is an image showing the attributes on features in the attribute table.



	fid	FACC	FSC	CMIX	CNAM	RTAI	LNAM	AO1	MODL	SSIZ
1	1	AL015	0	0		90		0	building_generic	30.060586603344586
2	2	AL015	0	0		53		0	building_generic	29.763286372603464
3	3	AL015	0	0		58		0	building_generic	28.513731120205716
4	4	AL015	0	0		90		0	building_generic	70.27184760382693
5	5	AL015	0	0		72		0	building_generic	22.75539982565757
6	6	AL015	0	0		90		0	building_generic	20.59044381714644
7	7	AL015	0	0		74		0	building_generic	84.18451379619785
8	8	AL015	0	0		54		0	building_generic	42.41597638906722
9	9	AL015	0	0		99		0	building_generic	29.712954603265997
10	10	AL015	0	0		66		0	building_generic	29.02103432759334
11	11	AL015	0	0		50		0	building_generic	29.254667804456794
12	12	AL015	0	0		65		0	building_generic	21.692016793416744
13	13	AL015	0	0		82		0	building_generic	32.403878762987 22
14	14	AL015	0	0		52		0	building_generic	29.717447262480768
15	15	AL015	0	0		97		0	building_generic	30.973428224488398

Figure 7. Attribute table for US CDB building data.

These point features were created using the following process:

1. Building footprints for the United States area were extracted from OpenStreetMap sources.
2. An approximate ‘significant size’ for each building we calculated using the formula: SignificantSize = sqrt($area) * 1.5
3. The centroid of each footprint polygon was generated, preserving attributes.
4. The centroid points were passed through an 'OSM to CDB Attribute Translator' to assign CDB-like attribution.

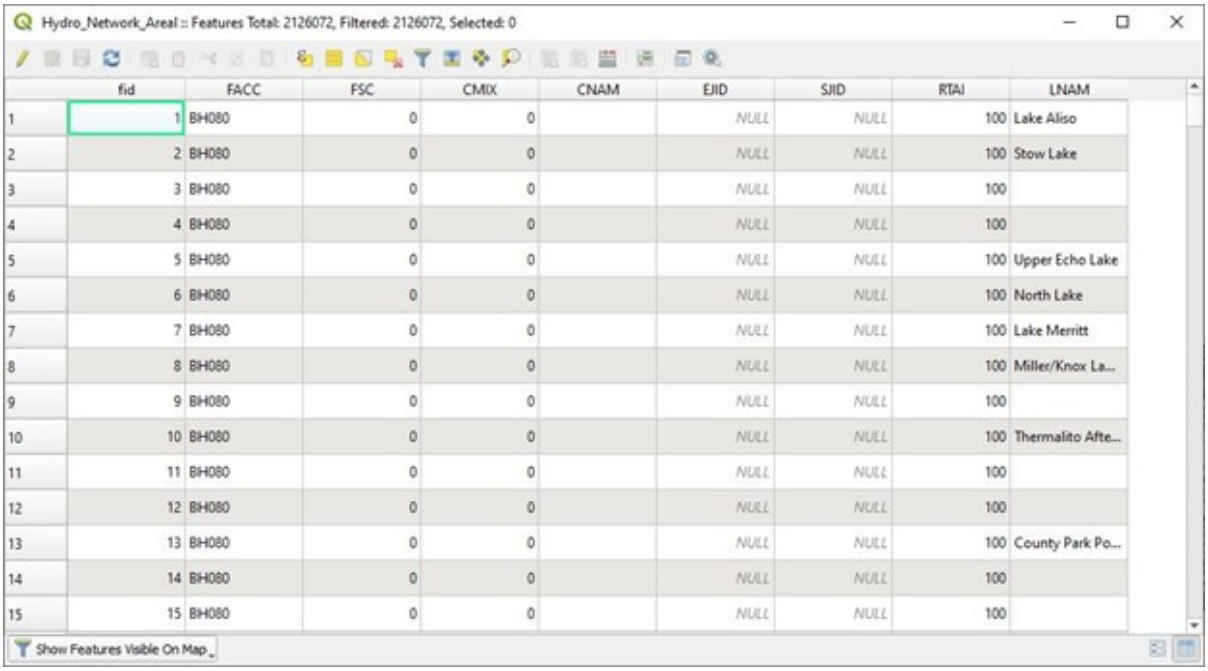
This data is contained in a layer named “GT_Models”, though the nature of the data can be considered as an agnostic representation of either GT or GS CDB models. Please note that the

original OSM building footprints do not have a ‘height’ attribute, so the derived significant size is approximate and present only for illustrative and comparative purposes.

9.4.2. Data Source 2 - osm-hydro-to-CDB.gpkg

This GeoPackage contains two layers with a total of 6,378,675 features and requires 5,043,081,216 bytes of storage.

The first layer is Hydro_Network_Areal, with attributes as shown below. This layer contains 2,126,072 features.

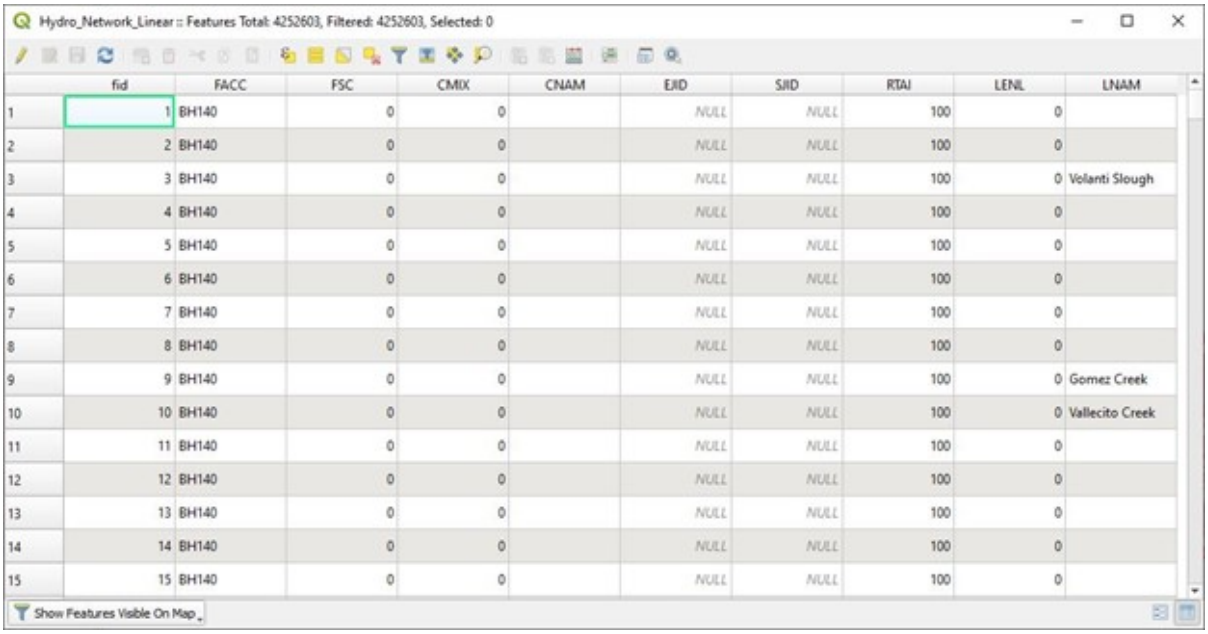


	fid	FACC	FSC	CMIX	CNAM	EJID	SJID	RTAI	LNAM
1	1	BH080	0	0		NULL	NULL	100	Lake Aliso
2	2	BH080	0	0		NULL	NULL	100	Stow Lake
3	3	BH080	0	0		NULL	NULL	100	
4	4	BH080	0	0		NULL	NULL	100	
5	5	BH080	0	0		NULL	NULL	100	Upper Echo Lake
6	6	BH080	0	0		NULL	NULL	100	North Lake
7	7	BH080	0	0		NULL	NULL	100	Lake Merritt
8	8	BH080	0	0		NULL	NULL	100	Miller/Knox La...
9	9	BH080	0	0		NULL	NULL	100	
10	10	BH080	0	0		NULL	NULL	100	Thermalito Afte...
11	11	BH080	0	0		NULL	NULL	100	
12	12	BH080	0	0		NULL	NULL	100	
13	13	BH080	0	0		NULL	NULL	100	County Park Po...
14	14	BH080	0	0		NULL	NULL	100	
15	15	BH080	0	0		NULL	NULL	100	

Figure 8. Hydro_Areal_Network CDB layer attribute table.

While named a ‘network’ layer, no effort was made to conduct a topological analysis and assign junction IDs. The CDB-like attribution is merely representative. This layer was created by combining OSM hydrographic polygons based on a very simple attribute filter, and then running the results through an 'OSM to CDB Attribute Translator’ with rules set to create very generic CDB attributions.

The second layer is ‘Hydro_Network_Linear, with attribution as shown below. This layer contains 4,252,603 features.



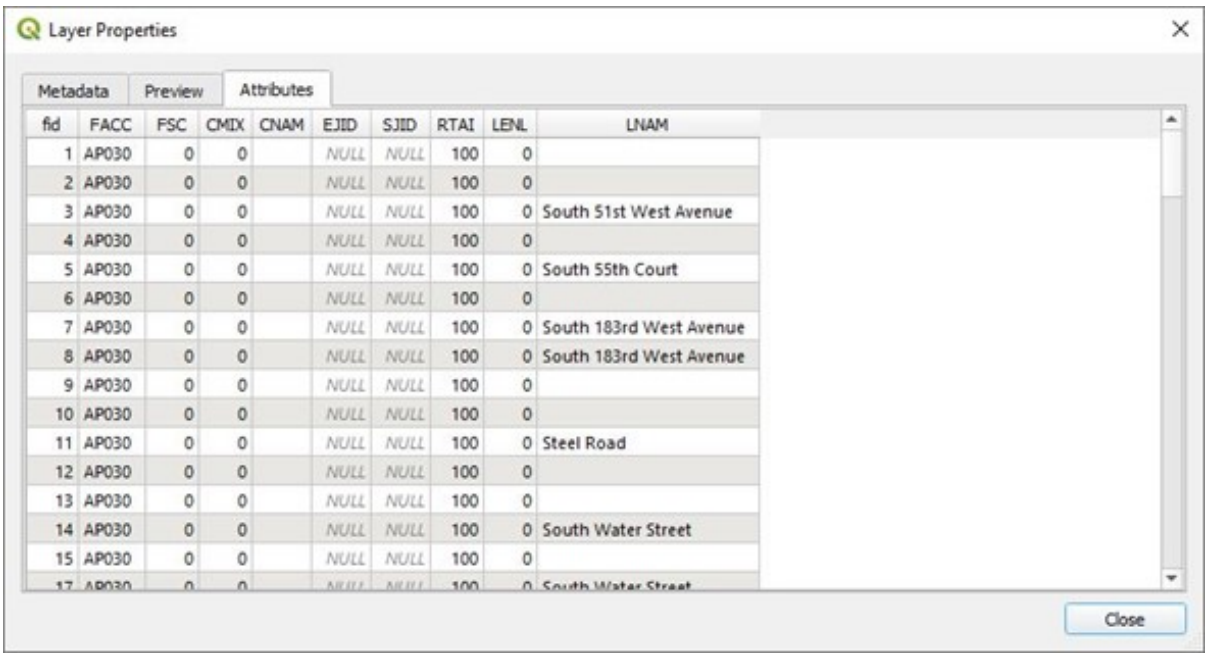
	fid	FACC	FSC	CMIX	CNAM	EJID	SJID	RTAI	LENL	LNAM
1	1	BH140	0	0		NULL	NULL	100	0	
2	2	BH140	0	0		NULL	NULL	100	0	
3	3	BH140	0	0		NULL	NULL	100	0	Volanti Slough
4	4	BH140	0	0		NULL	NULL	100	0	
5	5	BH140	0	0		NULL	NULL	100	0	
6	6	BH140	0	0		NULL	NULL	100	0	
7	7	BH140	0	0		NULL	NULL	100	0	
8	8	BH140	0	0		NULL	NULL	100	0	
9	9	BH140	0	0		NULL	NULL	100	0	Gomez Creek
10	10	BH140	0	0		NULL	NULL	100	0	Vallecito Creek
11	11	BH140	0	0		NULL	NULL	100	0	
12	12	BH140	0	0		NULL	NULL	100	0	
13	13	BH140	0	0		NULL	NULL	100	0	
14	14	BH140	0	0		NULL	NULL	100	0	
15	15	BH140	0	0		NULL	NULL	100	0	

Figure 9. Hydro_Network_Linear CDB attribute table.

Again, no effort was made to conduct a topological analysis and assign junction IDs. The CDB-like attribution is merely representative. This layer was created by combining OSM hydrographic linears based on a simple attribute filter, and then running the results through an 'OSM to CDB Attribute Translator' with rules set to create very generic CDB attributions.

9.4.3. Data Source 3 - osm-roads-to-CDB.gpkg

This GeoPackage contains roads derived from worldwide OSM. The GeoPackage contains 90,425,963 features and requires 29,832,347,648 bytes of storage.



fid	FACC	FSC	CMIX	CNAM	EJID	SJID	RTAI	LENL	LNAM
1	AP030	0	0		NULL	NULL	100	0	
2	AP030	0	0		NULL	NULL	100	0	
3	AP030	0	0		NULL	NULL	100	0	South 51st West Avenue
4	AP030	0	0		NULL	NULL	100	0	
5	AP030	0	0		NULL	NULL	100	0	South 55th Court
6	AP030	0	0		NULL	NULL	100	0	
7	AP030	0	0		NULL	NULL	100	0	South 183rd West Avenue
8	AP030	0	0		NULL	NULL	100	0	South 183rd West Avenue
9	AP030	0	0		NULL	NULL	100	0	
10	AP030	0	0		NULL	NULL	100	0	
11	AP030	0	0		NULL	NULL	100	0	Steel Road
12	AP030	0	0		NULL	NULL	100	0	
13	AP030	0	0		NULL	NULL	100	0	
14	AP030	0	0		NULL	NULL	100	0	South Water Street
15	AP030	0	0		NULL	NULL	100	0	

Figure 10. OSM roads layer attributes table.

Like the hydrographic features described above, this dataset does not contain a true network. Topology was not analyzed and CDB junction IDs are not set.

9.4.4. Output

The final GeoPackage container, combined-features-conus.gpkg, is simply a single container with each of the aforementioned layers copied into it. This GeoPackage requires 18,164,895,744 bytes of storage.

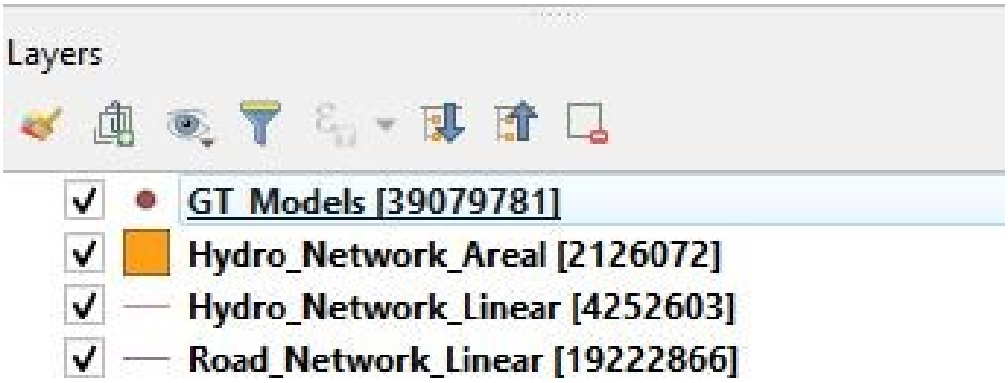


Figure 11. Layers of the final data container.

The layer 'Road_Network_Linear' was clipped to approximately the CONUS area from the world-wide road coverage. This is so this GeoPackage covers the same extents as the other three layers.

9.5. Performance Testing

9.5.1. Attribute Queries and Performance Summary

The objective of the testing was to explore a combination of spatial and attribution filtering in a CDB-like environment. To illustrate the importance of properly designing the schema when migrating from a Shapefile to a GeoPackage-based CDB, all the vector data in a CDB was converted. An approach similar to "Design Approach 4" in the OGC Discussion Paper titled OGC CDB, Leveraging GeoPackage Discussion Paper [https://portal.opengeospatial.org/files/?artifact_id=82553] was used. The conversion grouped all vector features by dataset and geocell into a single GeoPackage. Each vector feature was assigned a value for LOD, HREF, and UREF to correspond to the original Shapefile filename. A test was developed to randomly seek and read features in the CDB GeoPackage. The test script had a list of 8243 individual Shapefiles, but each file was opened and read in a randomized order. In the case of the Shapefile, each file was opened by filename, and all of the features were read. In subsequent tests with GeoPackage, the same content was read (using the list of Shapefile filenames), but instead of opening the Shapefile, the script performed a query based on the LOD, HREF, and UREF attributes.

In this test, reading the Shapefiles took 0:01:29 (1.5 minutes). With no indexes on the GeoPackage attributes, the queries took over one hour (1:01:47). Next, an index for the LOD, HREF, and UREF attributes was created and the above GeoPackage test repeated. With the indexes, finding and reading the same features took 0:00:49. This is only half of the time it took to read the Shapefiles.

9.5.2. Methodology

- The testing environment was a single Windows workstation, 16 CPU cores, 64 GB of system RAM, and very large SATA disk storage. No 'exotic' (SSD, M2, etc.) storage devices were used.
- Tests were created as Python scripts, leveraging the 'osgeo' Python module. Timing information was captured using Python's 'time' module. Python 3.7.4 (64-bit) was used.
- Each timing test was performed in the approximate CONUS extents of North 49 degrees latitude to South 24 degrees latitude, and from West 66 degrees longitude to West 125 degrees longitude.
- Prior to running a test, a 'step size' was defined – typically corresponding to a CDB LOD tile size. A list of every spatial filter in the entire extent was created then randomized.
- Also, prior to a test, a 'significant size' filter was set. When the layer 'GT_Model' is encountered, this additional attribute filter is applied. The intent is to mimic LOD selection, in addition to the spatial filter.
- There were three timing steps:
  - Timing step one is the elapsed time to apply the spatial filter.
  - Timing step two is the elapsed time to return a feature count based on the combined spatial and (if any) attribute filters.
  - Timing step three is the elapsed time to read the features from the layer into a Python list.
- At the end of processing and timing each 'tile' defined by the collection of spatial filters, a corresponding 'shape' is created and written into the test record output file.
- The output attribution is as follows:
  - count: the number of features returned after application of filters.
  - filter_t – time to complete the filtering operation(s) in seconds.
  - count_t: time to complete the feature count operation in seconds.
  - read_t : time to complete feature read operation in seconds. This includes reading from the GeoPackage container and appending each feature to a Python list.
  - Sequence: order that the tile was processed.
  - '$geometry': tile extents derived from spatial filter polygon.

Note: tiles that return zero features do not create a test output record.

9.5.3. Results

Experiment 1: Step size .25 degrees (CDB LOD2), significant size > 13.355 (LOD2 significant size)

Test results coverage is shown in the figure below.

Figure 12. Test results coverage at LOD 2.

This test simulates retrieving point features corresponding to CDB LOD2 and only models with the corresponding lowest significant size (as defined in the CDB 3.2, Table 3-1). The conclusion that is drawn from this test, however, is that spatial filtering time is insignificant and appears to not be correlated to the number of features found. The time taken to count and read filtered features appears to be a direct correlation to number of features found.

The Experiment 1 attribute table results are shown in the figures below, each filtered on a different field.

buildings-lod2-sim-reads :: Features Total: 11160, Filtered: 11160, Selected: 0

	count	filter_t	count_t	read_t	sequence
1	401983	0.000000000000...	0.859333753585...	3.327964305877...	20618
2	398348	0.000000000000...	0.906206846237...	3.374836683273...	23166
3	338184	0.000000000000...	0.796836376190...	2.812363862991...	16953
4	336415	0.000000000000...	0.705671072006...	2.608771085739...	1359
5	314047	0.000000000000...	0.843709230422...	2.734241008758...	6223
6	270986	0.000000000000...	0.578097581863...	2.265515327453...	13754
7	253266	0.000000000000...	0.640595436096...	2.187389850616...	21882
8	218113	0.000000000000...	0.281236648559...	1.624924659729...	9271
9	206850	0.000000000000...	0.296860218048...	1.531184196472...	14402
10	193259	0.000000000000...	0.352726936340...	1.436058759689...	2067
11	192505	0.000000000000...	0.406229972839...	1.453054666519...	23008
12	192471	0.000000000000...	0.296860218048...	1.421807289123...	7762
13	173336	0.000000000000...	0.312484741210...	1.359309196472...	14569
14	173111	0.000000000000...	0.343733549118...	1.359309434890...	22956
15	172134	0.000000000000...	0.656218051910...	1.593672990798...	12698

Show Features Visible On Map

Figure 13. Experiment 1 test results attribute table sorted by 'feature count'.

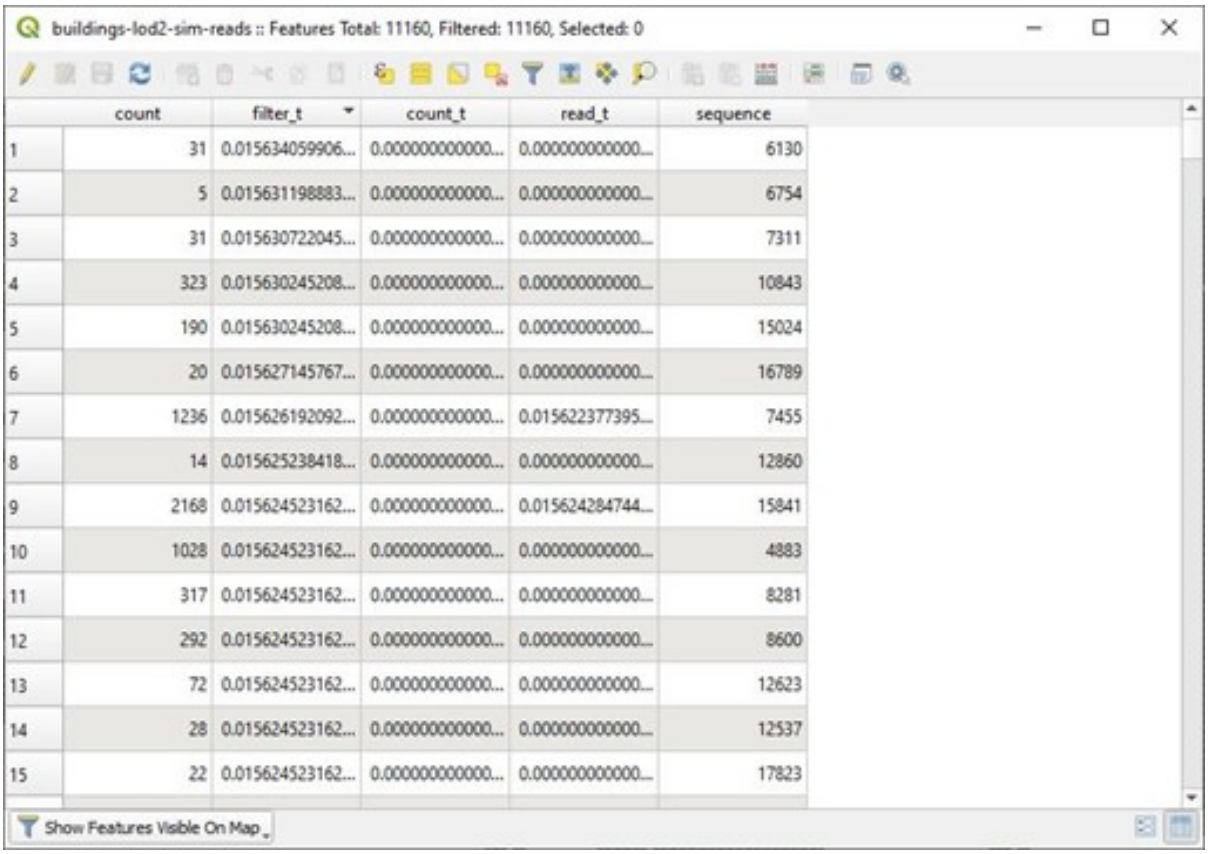


Figure 14. Experiment 1 test results sorted by 'filter_t'.

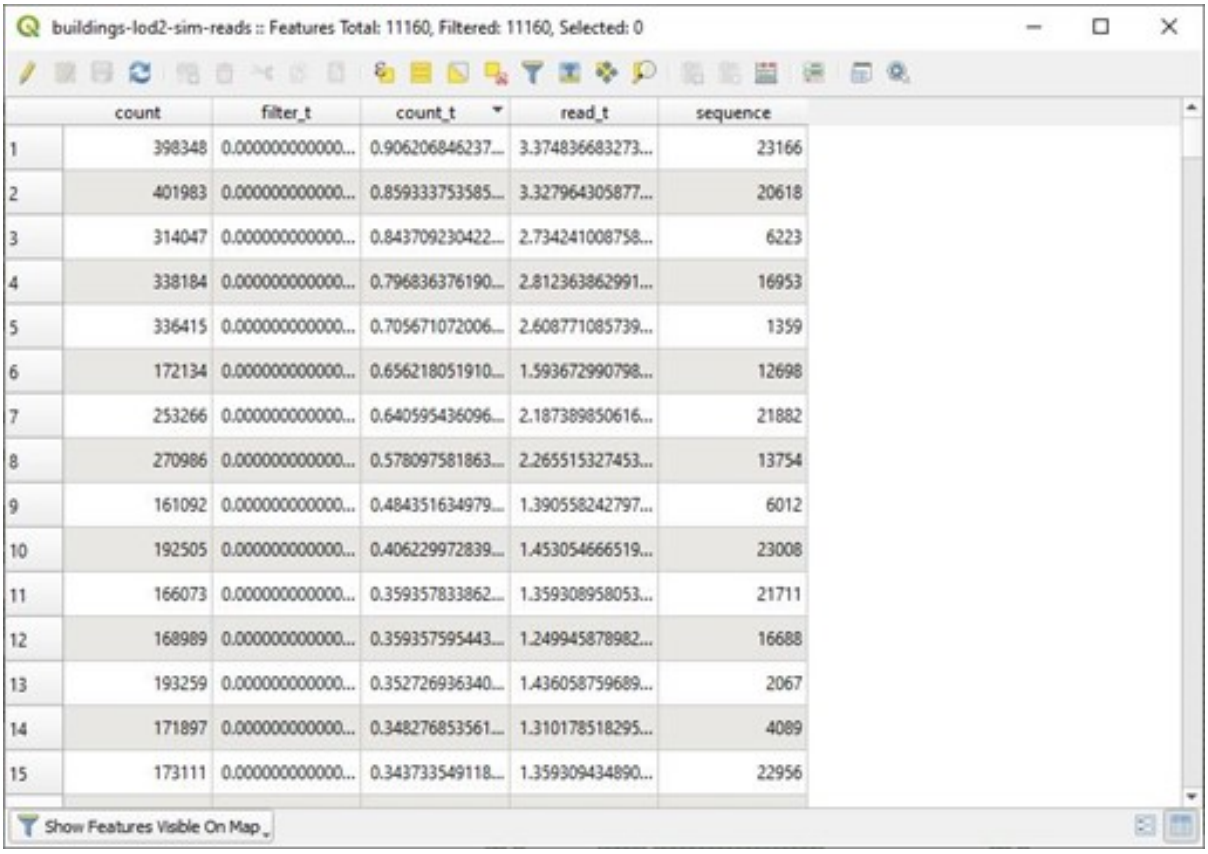
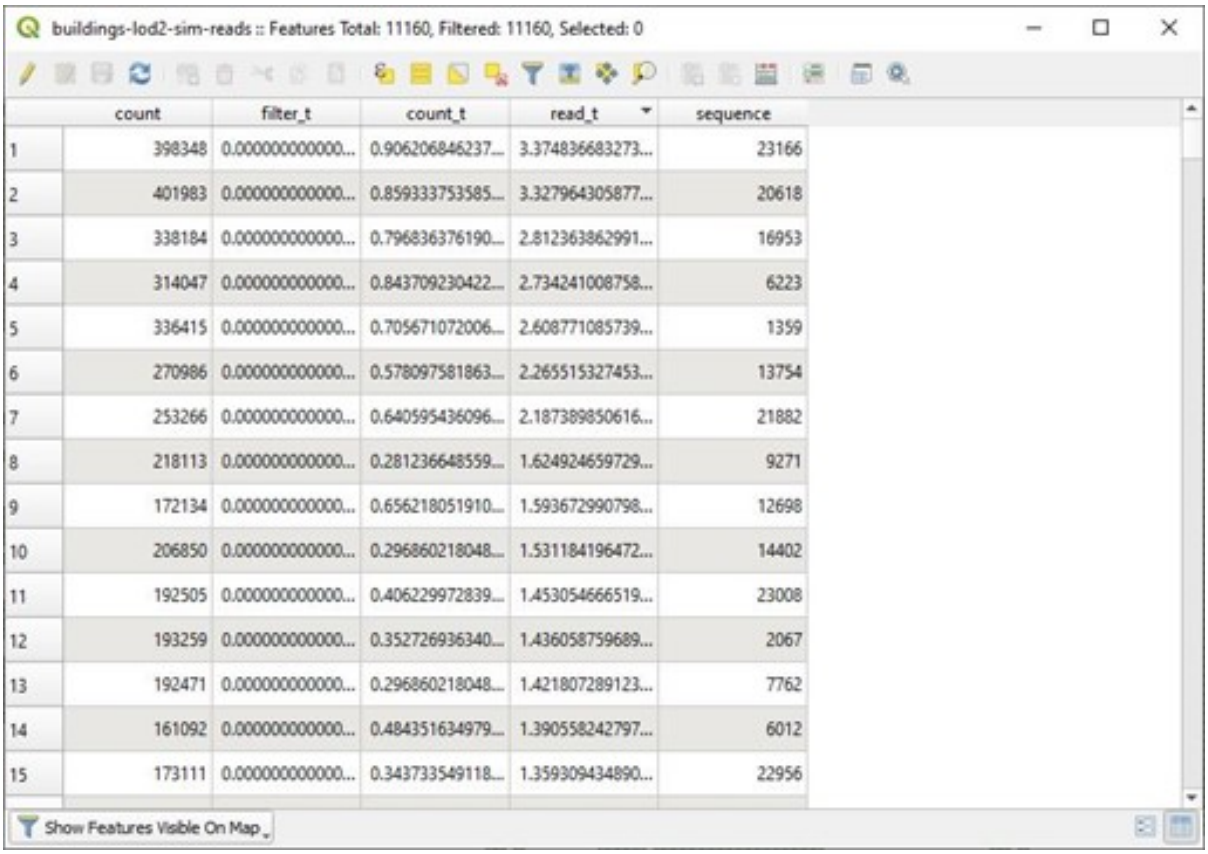


Figure 15. Experiment 1 test results sorted by 'count_t'.



	count	filter_t	count_t	read_t	sequence
1	398348	0.000000000000...	0.906206846237...	3.374836683273...	23166
2	401983	0.000000000000...	0.859333753585...	3.327964305877...	20618
3	338184	0.000000000000...	0.796836376190...	2.812363862991...	16953
4	314047	0.000000000000...	0.843709230422...	2.734241008758...	6223
5	336415	0.000000000000...	0.705671072006...	2.608771085739...	1359
6	270986	0.000000000000...	0.578097581863...	2.265515327453...	13754
7	253266	0.000000000000...	0.640595436096...	2.187389850616...	21882
8	218113	0.000000000000...	0.281236648559...	1.624924659729...	9271
9	172134	0.000000000000...	0.656218051910...	1.593672990798...	12698
10	206850	0.000000000000...	0.296860218048...	1.531184196472...	14402
11	192505	0.000000000000...	0.406229972839...	1.453054666519...	23008
12	193259	0.000000000000...	0.352726936340...	1.436058759689...	2067
13	192471	0.000000000000...	0.296860218048...	1.421807289123...	7762
14	161092	0.000000000000...	0.484351634979...	1.390558242797...	6012
15	173111	0.000000000000...	0.343733549118...	1.359309434890...	22956

Figure 16. Experimment 1 test results sorted by 'read_t'.

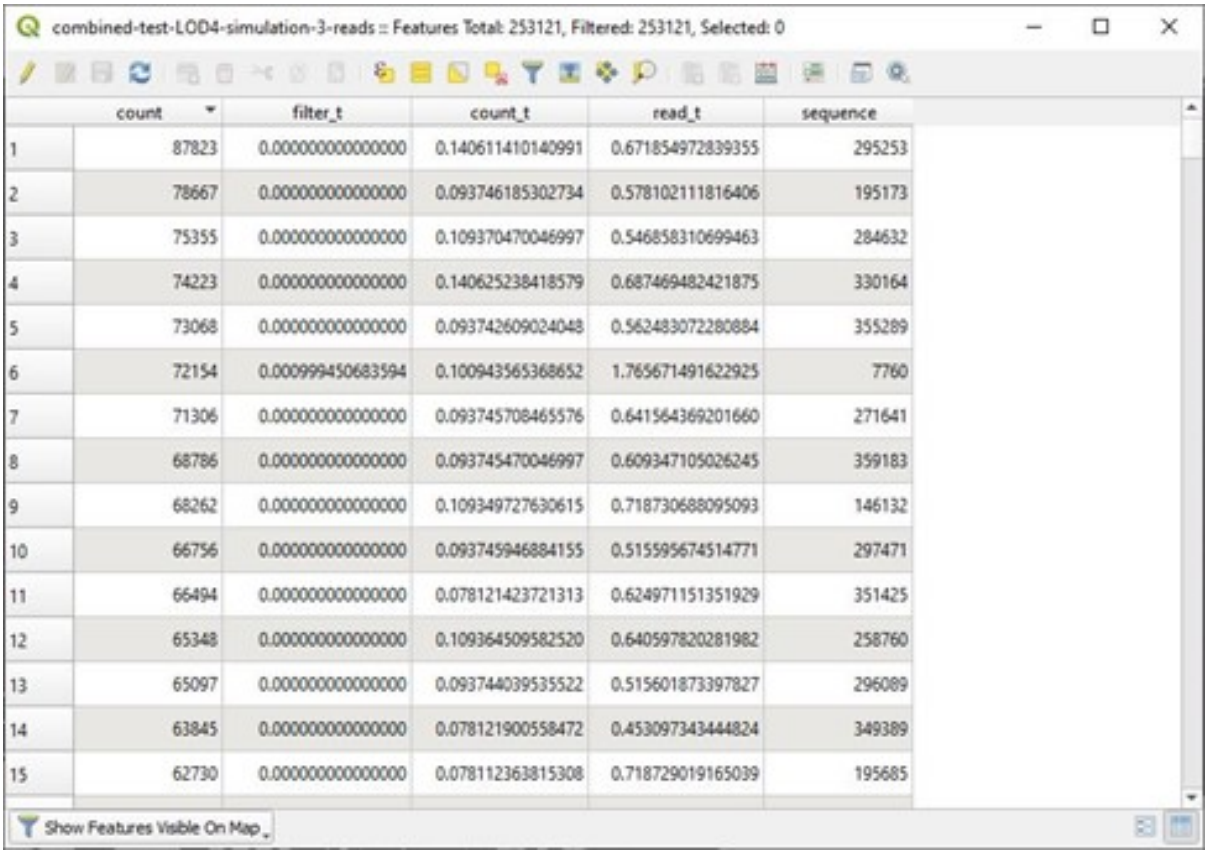
Experiment 2: Simulation of LOD4, hydro, road, and building layers, significant size (buildings) > 3.39718

Test results coverage is shown in the figure below.

Figure 17. Experiment 2 LOD 4 test coverage results.

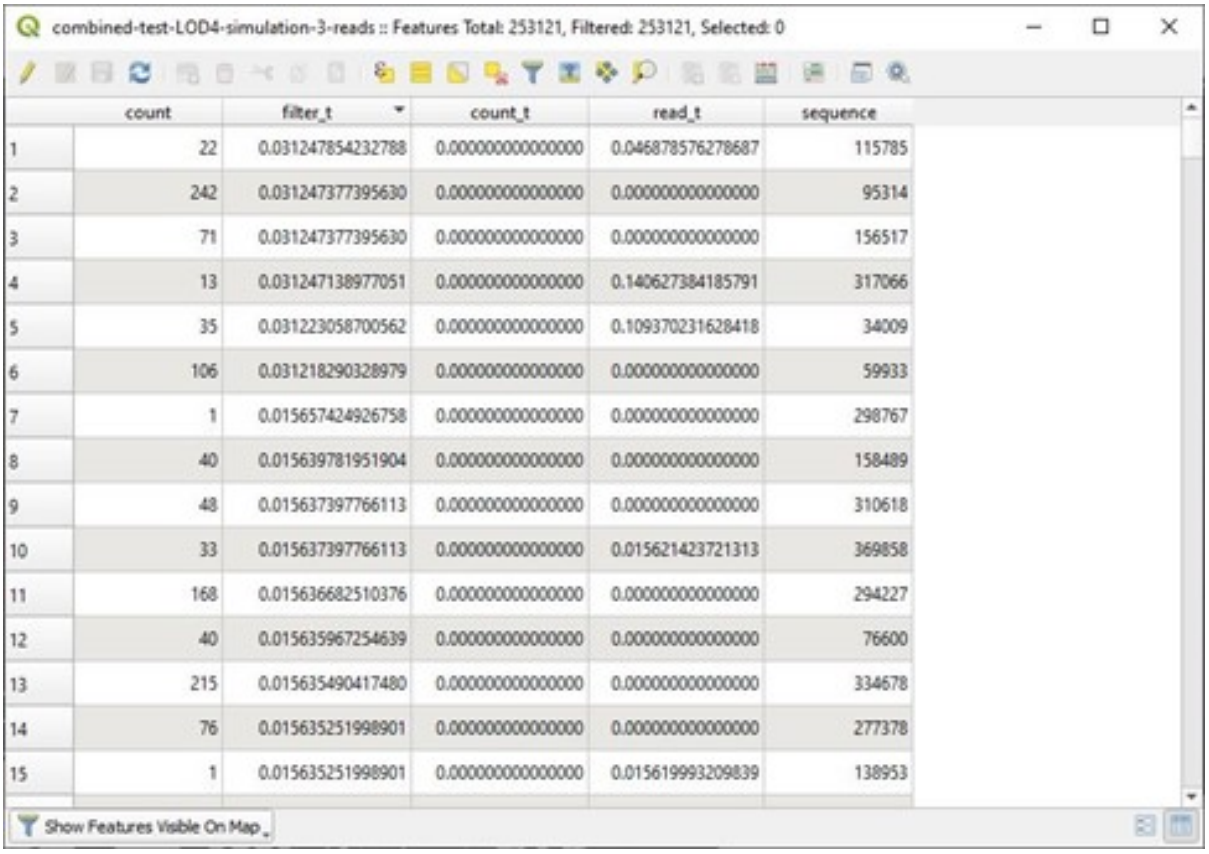
This test used the combined layers source file and simulates a CDB LOD4 data access pattern. Timing values are totals, accumulating as each layer was filtered, counted as features were read. Once again, filter timing appears to be insignificant and unrelated to the number of features filtered. Data in the GT_Model layer has both a spatial and attribute (significant size) filter applies.

The Experiment 2 attribute table results are shown in the figures below, each filtered on a different field.



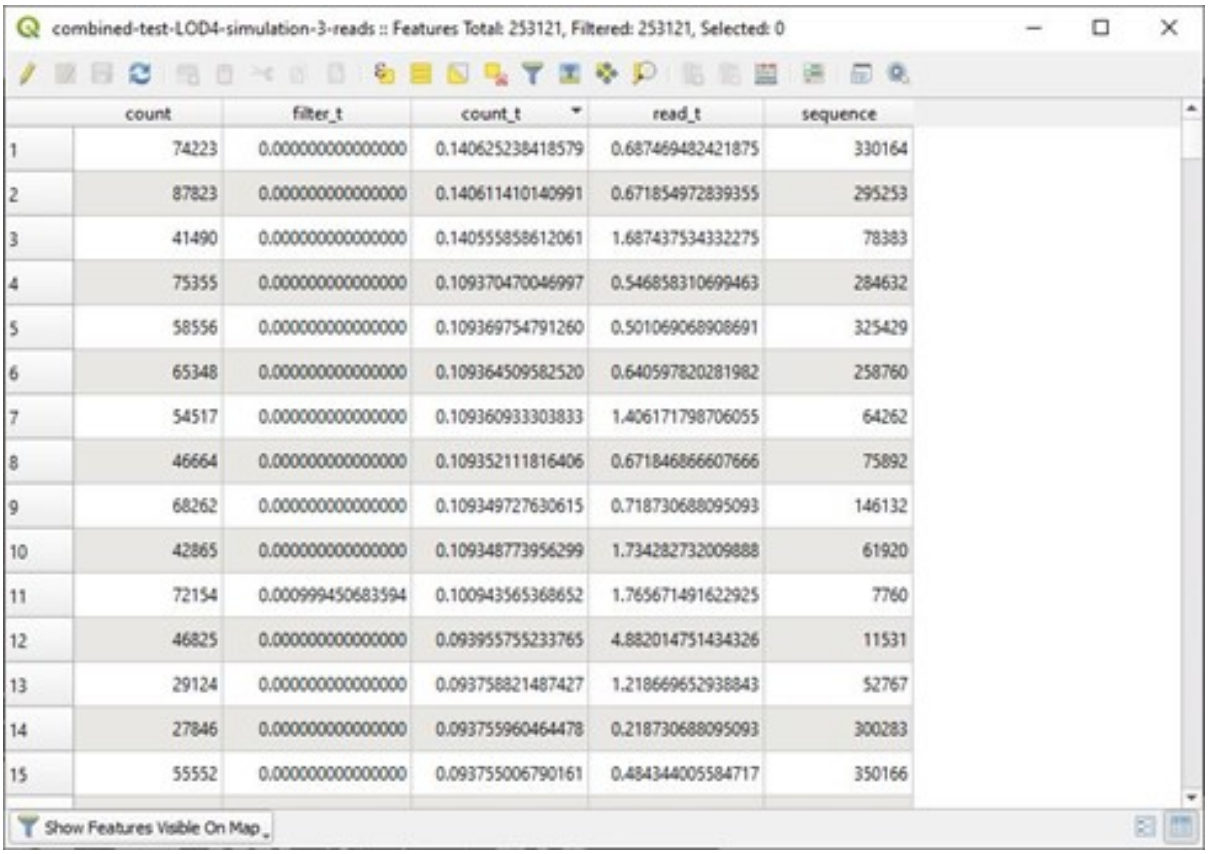
	count	filter_t	count_t	read_t	sequence
1	87823	0.000000000000000	0.140611410140991	0.671854972839355	295253
2	78667	0.000000000000000	0.093746185302734	0.578102111816406	195173
3	75355	0.000000000000000	0.109370470046997	0.546858310699463	284632
4	74223	0.000000000000000	0.140625238418579	0.687469482421875	330164
5	73068	0.000000000000000	0.093742609024048	0.562483072280884	355289
6	72154	0.000999450683594	0.100943565368652	1.765671491622925	7760
7	71306	0.000000000000000	0.093745708465576	0.641564369201660	271641
8	68786	0.000000000000000	0.093745470046997	0.609347105026245	359183
9	68262	0.000000000000000	0.109349727630615	0.718730688095093	146132
10	66756	0.000000000000000	0.093745946884155	0.515595674514771	297471
11	66494	0.000000000000000	0.078121423721313	0.624971151351929	351425
12	65348	0.000000000000000	0.109364509582520	0.640597820281982	258760
13	65097	0.000000000000000	0.093744039535522	0.515601873397827	296089
14	63845	0.000000000000000	0.078121900558472	0.453097343444824	349389
15	62730	0.000000000000000	0.078112363815308	0.718729019165039	195685

Figure 18. Experiment 2 LOD 4 test sorted by 'feature count'.



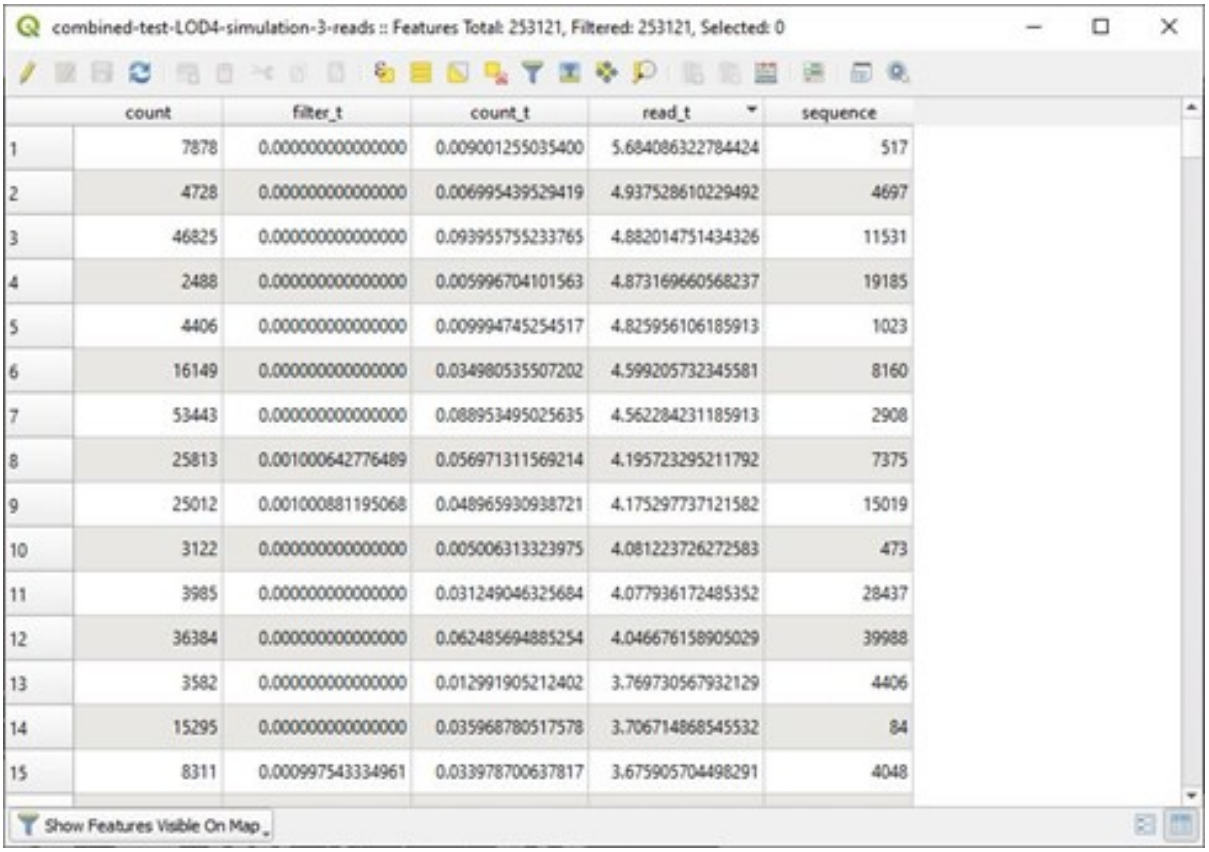
	count	filter_t	count_t	read_t	sequence
1	22	0.031247854232788	0.000000000000000	0.046878576278687	115785
2	242	0.031247377395630	0.000000000000000	0.000000000000000	95314
3	71	0.031247377395630	0.000000000000000	0.000000000000000	156517
4	13	0.031247138977051	0.000000000000000	0.140627384185791	317066
5	35	0.031223058700562	0.000000000000000	0.109370231628418	34009
6	106	0.031218290328979	0.000000000000000	0.000000000000000	59933
7	1	0.015657424926758	0.000000000000000	0.000000000000000	298767
8	40	0.015639781951904	0.000000000000000	0.000000000000000	158489
9	48	0.015637397766113	0.000000000000000	0.000000000000000	310618
10	33	0.015637397766113	0.000000000000000	0.015621423721313	369858
11	168	0.015636682510376	0.000000000000000	0.000000000000000	294227
12	40	0.015635967254639	0.000000000000000	0.000000000000000	76600
13	215	0.015635490417480	0.000000000000000	0.000000000000000	334678
14	76	0.015635251998901	0.000000000000000	0.000000000000000	277378
15	1	0.015635251998901	0.000000000000000	0.015619993209839	138953

Figure 19. Experiment 2 LOD 4 test sorted by 'filter_t'.



	count	filter_t	count_t	read_t	sequence
1	74223	0.000000000000000	0.140625238418579	0.687469482421875	330164
2	87823	0.000000000000000	0.140611410140991	0.671854972839355	295253
3	41490	0.000000000000000	0.140555858612061	1.687437534332275	78383
4	75355	0.000000000000000	0.109370470046997	0.546858310699463	284632
5	58556	0.000000000000000	0.109369754791260	0.501069068908691	325429
6	65348	0.000000000000000	0.109364509582520	0.640597820281982	258760
7	54517	0.000000000000000	0.109360933303833	1.406171798706055	64262
8	46664	0.000000000000000	0.109352111816406	0.671846866607666	75892
9	68262	0.000000000000000	0.109349727630615	0.718730688095093	146132
10	42865	0.000000000000000	0.109348773956299	1.734282732009888	61920
11	72154	0.000999450683594	0.100943565368652	1.765671491622925	7760
12	46825	0.000000000000000	0.093955755233765	4.882014751434326	11531
13	29124	0.000000000000000	0.093758821487427	1.218669652938843	52767
14	27846	0.000000000000000	0.093755960464478	0.218730688095093	300283
15	55552	0.000000000000000	0.093755006790161	0.484344005584717	350166

Figure 20. Experiment 2 LOD 4 test sorted by 'count_t'.



	count	filter_t	count_t	read_t	sequence
1	7878	0.000000000000000	0.009001255035400	5.684086322784424	517
2	4728	0.000000000000000	0.006995439529419	4.937528610229492	4697
3	46825	0.000000000000000	0.093955755233765	4.882014751434326	11531
4	2488	0.000000000000000	0.005996704101563	4.873169660568237	19185
5	4406	0.000000000000000	0.009994745254517	4.825956106185913	1023
6	16149	0.000000000000000	0.034980535507202	4.599205732345581	8160
7	53443	0.000000000000000	0.088953495025635	4.562284231185913	2908
8	25813	0.001000642776489	0.056971311569214	4.195723295211792	7375
9	25012	0.001000881195068	0.048965930938721	4.175297737121582	15019
10	3122	0.000000000000000	0.005006313323975	4.081223726272583	473
11	3985	0.000000000000000	0.031249046325684	4.077936172485352	28437
12	36384	0.000000000000000	0.062485694885254	4.046676158905029	39988
13	3582	0.000000000000000	0.012991905212402	3.769730567932129	4406
14	15295	0.000000000000000	0.035968780517578	3.706714868545532	84
15	8311	0.000997543334961	0.033978700637817	3.675905704498291	4048

Figure 21. Experiment 2 LOD 4 test sorted by feature 'read_t'.

9.6. Conclusions and Recommendations

1. Storing large amounts of feature data in single GeoPackage containers and retrieving that data by applying spatial and attribution filters that correspond with typical CDB access patterns appears to be practical.
2. Spatial filters easily mimic the existing CDB tiling scheme.
3. Storing ‘significant size’ on model instancing point features can significantly improve the model retrieval scheme, rather than storing models in the significant size related folder scheme. Storing and evaluating significant size on instancing points can make visual content and performance tuning much more practical.

Chapter 10. 3D Models and other 3D Content

This section discusses the proposal to use glTF to encode and store 3D models in the CDB X data store. This section also describes the prototype testing performed during the the <activity>. In CDB, 3D models are such phenomenon as trees (static), buildings, transmission towers, or helicopters (dynamic).

Currently, the OpenFlight specification is used to define all the 3D models in a CDB data store. The current OGC CDB 1.x core standard and Volume 6 CDB Rules for Encoding Data using OpenFlight [https://portal.opengeospatial.org/files/16-009r4] specifies all of the rules and requirements for using OpenFlight models in a CDB data store. The early advantage OpenFlight held over many 3d geometry model file formats (.obj, .dxf, .3ds) was its specific real-time 3d graphics industry design.

For CDB X, the participants in the 3D Geospatial Series Tech Sprint II – OGC CDB 2.0 activity agreed that a more modern 3D model encoding and transmission format should be explored. Based on wide adoption and use in many domains, the Khronos glTF 2.0 specification was selected for further research and prototyping. A key part of this activity was to perform a detailed analysis of glTF functionality as compared to OpenFlight capabilities. The participants agreed that no (or as little as possible) functionality should be lost. NOTE: glTF will not be a replacement for OpenFlight in a CDB data store. Instead, glTF is recommended as the preferred encoding for CDB 3D Models. There are a few OpenFlight to glTF conversion tools.

The remainder of this section discusses the findings of the comparison as well as lessons learned from the prototyping activity.

10.1. Questions to be addressed

Following are some of the questions and suggestions for experimentation that the 3D Modelling Subgroup initially identified as needing further discussion and possible experimentation. These formed the basis for the specific questions and approaches uses in the CDB-X 3D Prototyping and Experimentation.

1. Models be stored as files such as OpenFlight or in GeoPackage.
2. Models preferred to be glTF (Later consider I3S, 3D Tiles, CityGML and other encodings)
3. Textures shall be xxxx format (png, jp2) Power of 2 enforced? {**Not sure what this means**}
4. UV Map with model?
5. Naming convention tbd - should have some metadata use - indexing? directory path? layer? featureID?
6. LoD Structure: MS LoD extention for glTF valid for CDB work? Model LoDs have vertex count, significant size etc… and have their own LoD table which is not the same as tiled LoDs.
7. Describe model placement.
    a. Point features
    b. Terrain Reconstruction
    c. Elevation

d. Imagery
e. Polygon features (Formerly aerial features in CDB 1.1 and earlier).
f. Linear features

8. Is there a way to map glTF Material vs CDB physical material?

CDB catalog service for 3D tile inside a tile

10.2. CDB X Requirements for 3D Model Components

Based on the questions and follow-up discussion, the CDB X participants derived the following requirements for CDB-X model components.

Requirement	**Objectives**
Improve interoperability with OWT	* Align 3D model geometry attribution in order to support conversion between the two standards. Test that conversion in both directions. * Support similar encoding (glTF) to support convergence as standards evolve.
Serve archiving/repository use case and runtime usage	* Package 3D models LODs in an easy to edit manner while allowing fast runtime access of independent LODs. * Increase 3D tools (edit and view) interoperability. * Support multiple encoding with metadata to identify which one is used (Starting with OpenFlight and glTF).
Modernize CDB	* Support newer 3D encoding formats (glTF initially) * Support newer attribution (NAS/GGDM) in point feature and in 3D geometry * Provide flexibility to extend 3D models components and attribution * Review LOD “bins” in context of today’s GPU capabilities * Added constructs for better game engine support (navigation and collision mesh, PBR etc..) *Review lightpoint (Dataset and in 3D model), light source, lightmap.
Support SOF use cases : Same dataset for M&S and C2 including the mission command and TAK devices	* ATAK: Support 3D view and attribution * ATAK: Support building interior 3D encoding (and 2D floorplan?) * ATAK: Support packing 3D models in small memory footprint * ATAK: Support LOD selection “per model or per zone” to transfer only the required data * Mission command: Support merging of CDB (Versioning) to cover large area * Mission command: Support 3D WEB based map steaming

10.3. Short and Medium Term Objectives

The following are the short and medium term objectives for the 3D Model Prototyping and Experimentation. These design objectives are based on the initial set of questions posed by the CDB-X participants.

1. Support more 3D model encoding with metadata detailing which is used. Preserve OpenFlight (and potentially improve its usage), add glTF with extensions
2. Reduce file count and package models better while providing enough indexing data to allow readers to quickly extract only what they need. Metadata/catalog - support editing/replace use case
3. Address GS, GT, model instancing
4. Support Building interior with navigation mesh and collision mesh
5. Encode 3D models in glTF
    a. Node grouping, LOD grouping in json
    b. Define CDB extensions for glTF
    c. Look at tools to allow CDB attribution editing
6. Model packing with MetaData

10.4. 3D Experiment Object Model

10.4.1. The data

This experiment uses a GeoPackage comprised of a point feature table with attribution.

```
sqlite> select * from ?????;
id  A01       CNAM                      RTAI  AHGT   BBH     BBL     BBW     BSR     CMIX  CNAM                      FACC  FSC HGT   MLOD  MODL                  NIS NIX NNL NTC NTX NVT SSC SSR
--- --------- ------------------------- ----- ------ ------- ------- ------- ------- ----- ------------------------- ----- --- ----- ----  --------------------- --- --- --- --- --- --- --- ---
1   12.3653   AL241000-39-12U12R39-0    0     FALSE  15.624  40.598  35.833  28.179  2     AL015000-38-12U12R38-0    AL015 0   15.62 2     Building_-5616_-12643 1   0   0   0   0   225 0   0
```

Figure 22. 3D Experiment 1 attribute table for point features.

10.4.2. Attribute changes explored

The following current CDB attribute changes were explored.

- CNAM - no longer needed
- FACC - FSC - Change to GGDM - impact on field type
- MLOD - (LOD of the model to use) This is closely linked to tiling and LODs of vector where each point features would point to a model at a given LOD (should be lower than feature vector LOD). Presume at the moment we keep this.
- MODL - (Name of the model to use). In this case, we should point to a metadata file for the model. Alternatively, the content of the metadata could be encoded in GeoPackage - which is bad for tool operability.

A key challenge to resolve is how to link the model data with the feature data. Currently, the MLOD

and MODL are composed and lookup into the tile index of the point feature. Is that functional capability (approach?) preserved? Do we use a model table in Geopackage with a primary key? As decided, models would remain separate files and not encoded in the GeoPackage. So, a unique file name is required for this experiment.

10.5. Description of 3D formats/encodings and References for this section:

10.5.1. OpenFlight

OpenFlight (or .flt) is an open, freely available 3d geometry model file format originally developed by Software Systems Inc. for its MultiGen real-time 3d modeling package and now actively maintained by the OGC member Presagis [https://www.presagis.com/en/]. OpenFlight is an open format, binary encoded with support for user extensions, which is supported widely in modeling and simulation community for dynamic and static 3D model. OpenFlight has numerous constructs that have no equivalent to date in other open standards. In CDB 1.x, OpenFlight is used for the representation of 3D static and dynamic models and RGB format for the 3D model's textures. OpenFlight is now an OGC Community standard.

OpenFlight v16 [https://portal.opengeospatial.org/files/90663]

Recommendation

```
The 3D model group recommends that a new CDB standard adopts the latest version of
OpenFlight,
leveraging the extended materials, hot spots and other important constructs it brings.
```

10.5.2. GL Transmission Format (glTF) 2.0

glTF is a royalty-free specification for the efficient transmission and loading of 3D scenes and models by applications. glTF uses the JSON standard. glTF is an API-neutral runtime asset delivery format developed by the Khronos Group 3D Formats Working Group.

glTF 2.0 [https://github.com/KhronosGroup/glTF/tree/master/specification/2.0] GitHub repo and description.

KHR_lights_punctual [https://github.com/KhronosGroup/glTF/tree/master/extensions/2.0/Khronos/KHR_lights_punctual] is an extension that defines a set of lights for use with glTF 2.0. Lights define light sources within a scene.

MSFT_lod [https://github.com/KhronosGroup/glTF/tree/master/extensions/2.0/Vendor/MSFT_lod] is an extension that adds the ability to specify various Levels of Detail (LOD) to a glTF asset.

EXT_mesh_gpu_instancing [https://github.com/KhronosGroup/glTF/tree/master/extensions/2.0/Vendor/EXT_mesh_gpu_instancing] is an extension that is specfically designed to enable GPU instancing which renders many copies of a single mesh at once using a small number of draw calls. It's useful for things like trees, grass, road signs, etc.

FB_geometry_metadata [https://github.com/KhronosGroup/glTF/tree/master/extensions/2.0/Vendor/

FB_geometry_metadata] is an extension that annotates glTF scene objects with a summary of the cumulative geometric complexity and scene-space extents of the scene's associated scene graph. Editors note: While the computed total vertex and primitive count are metadata this is very limited metadata and may not meet the needs of the CDB X community.

KHR_materials_unlit [https://github.com/KhronosGroup/glTF/tree/master/extensions/2.0/Khronos/KHR_materials_unlit] is an extension that defines an unlit shading model for use in glTF 2.0 materials, as an alternative to the Physically Based Rendering (PBR) shading models provided by the core specification.

KHR_texture_transform [https://github.com/KhronosGroup/glTF/tree/master/extensions/2.0/Khronos/KHR_texture_transform] is an extension that adds offset, rotation, and scale properties to textureInfo structures. These properties would typically be implemented as an affine transform on the UV coordinates.

AGI_articulations [https://github.com/KhronosGroup/glTF/tree/master/extensions/2.0/Vendor/AGI_articulations] is an extension that adds articulations that define the names and ranges of allowable motions of nodes on a models. The extension includes articulation stages, pointing vectors, and attach points.

10.5.3. Original Prototyping Experiments - Keeping for now but will delete when this section is more mature.

Two profiles.

Profile 1 Storing the mesh in as 3D tile: Experiments

1. Question: How to store LODs? In glTF extensions, In 3D tiles, in separate glTF
2. Question: Are there missing glTF constructs?

Profile 2 Storing the mesh in as 3D tile (Editor's note: Why just 3D Tiles? Why not a more general approach that allows other encodings/approaches?)

1. Question: What is the ability to source edit the mesh?
2. Question: Mixing GeoPackage with 3D tiles?
3. Question: CDB catalog service for 3D tile inside a tile

Both Profile 1 and Profile 2

1. Question: What are the efficient format/packing for each different end-points:
    a. ATAK
    b. Mission command (wms/wfs)
    c. Modsim
    d. Gaming

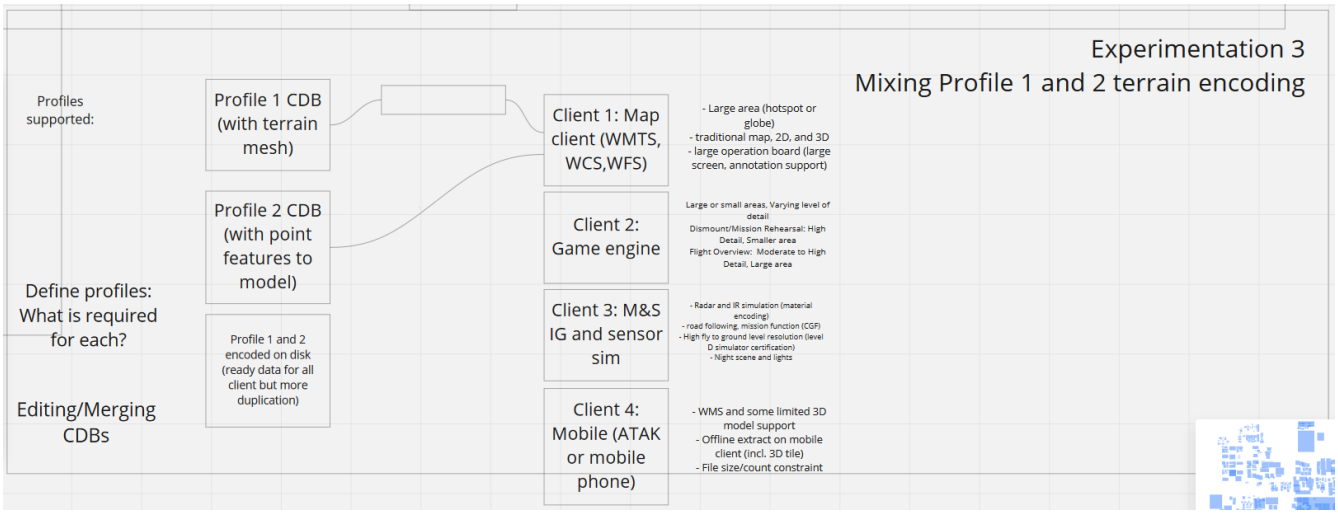


Figure 23. Profles and Experiments for CBD-X 3D Models.

10.6. Investigation 1 - Make CDB 3D models easier to maintain and/or modify

One of the objective of the 3D model refactoring in CDB is to facilitate the editing {**Update**} process. In the CDB 1.X environment, 3D model encoding is hard because: . The OpenFlight format has fewer editing tools than in newer 3D formats {**Give an example**}. This situation led the 3D subgroup to investigate the glTF encoding for possible use in CDB-X. . The CDB 1.X model components (geometry, texture, metadata) as well as the Level of Detail (LoD) for those components are stored in different folders. This distribution is not well supported by editing tools and edits often require updating many files, including collections contained in Zip files. Therefore the 3D subgroup investigated a grouping for all 3D model components.

While investigating these potential solutions, understand the motivationing and design requirements behind the original CDB structure is required so as to properly assess the impact of the changes against the original design criteria. This would include a determination on the current installed CDB user base.

CDB 1.X Geospecific 3D models were likely motivated by:

1. A single encoding (OpenFlight) for 3D model geometry: At the time of the development of the original CDB specification, OpenFlight was the most adopted and used format for 3D models in the Modeling and Simulation industry. OpenFlight meet the requirements for an open format, an efficient binary encoding, and an editable source format supported by a number of tools. Having a single possible encoding increases interoperability by reducing the number of encoding to support on both the CDB producer and the consumer side. Supporting different encodings (OpenFlight and glTF) may result in less interoperability. In some cases, multiple versions of a CDB with the different encodings may be required as different consumers may only support one encoder.
2. Facilitate parallel access to all components as opposed to sequential access. By grouping all components into datasets (textures geometry, material etc...) into tiles and LODs, a client can assess which tile and component is required and fetch all components in parallel. In other words, without having to read the result of a request before sending a new request (like waiting for geometry to identified {**Confusing here. Not sure what to write**} referred textures and

materials).

Focusing on GeoSpecific models

The initial experimentation targeted Geospecific models inside the CDB structure. This was because they represent the greatest challenge. The GeoTypical models still need to be reviewed. However, the belief is that a sub-set of the GeoSpecific concepts can also be applied to GeoTypical models. The tasks describe below document the investigation along with the results:

10.6.1. TASK 1 - CDB OpenFlight and glTF Feature Analysis

OGC CDB 1.X relies on a number of OpenFlight constructs that have no equivalent in glTF. One of the first task of the group was to list all OGC CDB 1.X requirements with regard to 3D model encodings and assess whether glTF has equivalent constructs. The following provides details of the resultes of the analysis.

==

OpenFlight Construct	CDB Requirement/Guidelines (all from Chapter 6)	Notes	glTF Equivalent	New glTF extension required	Mapping OF to glTF difficulty	Implemented in converter prototype
Node Hierarchy	Req 2 - Global Zone Req 3 - Model Zone Chap 6.5.6.1 - Model Landing Zone Req 17 - Model Footprint Zone Req 18 - Model Footprint Hierarchy Req 19 - Model Cutout Zone Req 20 - Model Cutout Geometry Req 21 - Model Pseudo-Interior Zone Req 22 - Model Interior Zone Req 4 - Layers	Chap 6.2 - Using hierarchy to organize models helps identify components of intereset	Node Hierarchy fully supported	No	Easy	Yes

OpenFlight Construct	CDB Requirement/Guidelines (all from Chapter 6)	Notes	glTF Equivalent	New glTF extension required	Mapping OF to glTF difficulty	Implemented in converter prototype
Comment Attribute	Req 14 - Zone Names Chap 6.5.3.1 - Zone Material Index Req 16 - Hot Spot Temperature Req 23 - Model Point (Damage) Req 24 - Model Point (DIS Origin) Req 25 - Model Point (View Point) Chap 6.6.2.3 - Model Attach Point Chap 6.6.2.4 - Model Anchor Point Chap 6.6.2.5 - Model Center of Mass Req 29 - Damage State (Levels) Req 36 - Model Vendor Attributes	Comment field is used to extend OpenFlight node for CDB specialization for many uses A number of constructs are available in the OF spec that could be natively encoded in the format instead of using the comment field.	Implemented using CDB extension	Yes	Easy	Yes

OpenFlight Construct	CDB Requirement/Guidelines (all from Chapter 6)	Notes	glTF Equivalent	New glTF extension required	Mapping OF to glTF difficulty	Implemented in converter prototype
Node Attribution	Req 13 - Model Zone Bounding Box	Group node has bounding box attribute Other attributes available on the node: Roof Flag Culture Footprint	Primitives have min/max properties. For general nodes, this must be stored using the CDB extension	Yes	Easy	Yes
External Referencing	Chap 6.2.4 - Model Master File Req 6 - Use XRef node to reference other files Chap 6.2.5.1 - Models straddling multiple files Chap 6.2.5.2 - Models with multiple LODs	Model Master File (comprised of xrefs) ensures a Model is seen as a single "object" even though its constituents are stored in separate files	No external referencing	Yes	Difficult	Partial The converter can parse external references and combine them into one gltf file (using mesh instancing for repeated references)

OpenFlight Construct	CDB Requirement/Guidelines (all from Chapter 6)	Notes	glTF Equivalent	New glTF extension required	Mapping OF to glTF difficulty	Implemented in converter prototype
Level of Detail Node	Chap 6.2.5.2 - Models with multiple LODs Req 26 - Significant Size Chap 6.8.1 - Exchange LODs Chap 6.8.2 - Additive LODs	LOD node contained in XRefs in master file Significant Size attribute on LOD LOD node support exchange and additive LOD strategies	Requires extension (MSFT_lod) No sig size. Works with switch in/out	Yes	Medium	Yes Uses MSFT_lod extension
Switch Node	Req 27 - Switch Masks (one per state) Req 28 - Switch Mask Names Req 29 - Damage State (Transition) Req 31 - Blur State (Transition)	Switch node supports multiple masks. Each mask can be named.	No native support for this.	Yes	Medium	Yes

OpenFlight Construct	CDB Requirement/Guidelines (all from Chapter 6)	Notes	glTF Equivalent	New glTF extension required	Mapping OF to glTF difficulty	Implemented in converter prototype
Degree of Freedom Node	Req 32 - Articulation Req 33 - Gimbal Limits	DOF node supports min/max limits for each degree of articulation (translation, scale, rotation)	Skins, Joints, Animations. glTF is more flexible/complex than OF when it comes to animated models	No	Medium	
Light Point Node	Req 35 - Model Light Points	Light Point node can represent individual points or light "string"	No native support. Existing extensions for light sources but these are a different concept than light points	Yes	Difficult	

OpenFlight Construct	CDB Requirement/Guidelines (all from Chapter 6)	Notes	glTF Equivalent	New glTF extension required	Mapping OF to glTF difficulty	Implemented in converter prototype
Projection	Req 1 - Specify Projections	Required projections for GTModel, GSModel, MModel and T2DModel	Could be done as part of the CDB extension or using a separate extension to improve interoperability. Technically not required for CDB because CDB enforces the projection of models. GSModels, GTModels and MModels must use flat earth and T2DModels must use geodetic.	No	Medium	

OpenFlight Construct	CDB Requirement/Guidelines (all from Chapter 6)	Notes	glTF Equivalent	New glTF extension required	Mapping OF to glTF difficulty	Implemented in converter prototype
Coordinate System	Req 7 - X (left/right), Y (front/back), Z (bottom/top) Req 8 - Origin (0,0,0)	These are native OpenFlight conventions	glTF 2.0 uses a right-handed coordinate system, with x point right, y point up and z backward Changing this requires an extension and would reduce performance and interoperability Would recommend keeping glTF's axis system and adjusting the standard if needed.	No	N/A	
Local Coordinate System	Chap 6.3.1.2	Transformation Matrix is used to specify LCS	Transformations on nodes	No	Easy	Yes
Units	Char 6.3.1.3	Header attribute is used to specify Units	Can be specified in extras property	No	Easy	

OpenFlight Construct	CDB Requirement/Guidelines (all from Chapter 6)	Notes	glTF Equivalent	New glTF extension required	Mapping OF to glTF difficulty	Implemented in converter prototype
Instancing	Req 11 - Avoid repeating identical pieces of geometry	Efficiency - smaller database size	Multiple nodes can instantiate the same mesh. However, there is no concept of node instancing. OF is more flexible	No	Medium	Yes
Mesh	Req 11 - Favor mesh over polygons	Efficiency - smaller database size, fewer graphics states	Mesh is supported and highly recommended over polygons. In OF many models use individual polygon nodes, but this would be inefficient in glTF. May lead to large geojson files.	No	Easy	Yes Polygon nodes at the same level with the same textures are merged into one mesh
Vertex Ordering	Req 11 - CCW order of verts define polygon "front"		GLTF uses CCW ordering of vertices	No	Easy	Yes

OpenFlight Construct	CDB Requirement/Guidelines (all from Chapter 6)	Notes	glTF Equivalent	New glTF extension required	Mapping OF to glTF difficulty	Implemented in converter prototype
Relative Priority	Req 12 - Layers of coplanar geometry	Relative Priority attribute at : • Face • Mesh • Object • Group	Not supported natively. Can be stored in an extension, but per-face priority is more complicated. Two possible solutions : • Faces with different priorities should be grouped in separate meshes • Use an extension to support per-face metadata	No	Easy	

OpenFlight Construct	CDB Requirement/Guidelines (all from Chapter 6)	Notes	glTF Equivalent	New glTF extension required	Mapping OF to glTF difficulty	Implemented in converter prototype
Textures	Req 37 - Textures stored in separate files from models Req 41 - Relative Texture Paths Req 42 - Object Shadow Attribute	Loading efficiency	Textures supported. Materials in glTF are similar to extended materials in OF, but not all layers from openflight exist in glTF. Ex: Light map, specular map, reflection map. Material textures are not a concept in gltf. Would require extension.	Yes	Difficult	Yes

==

==

==

==

Chapter 11. Supporting more than CDB 1.X:

OpenFlight capabilities that could be leveraged:

- Extensions
- Extended Materials
- Hotspots
- LOD Transitions
- Cultural Footprint
- Point Nodes (Model Points instead of using tranforms)

11.1. TASK 2 - Developing glTF CDB extensions

`EXT_CDB` is a draft glTF extension that adds required CDB simulation capabilities identified as missing from glTF. The new (and prototype) capabilities include:

- Indicating that a glTF node is a certain type of CDB point: Attachment point, anchor point, center of mass, DIS origin, or viewpoint.
- Referencing light points from glTF nodes.
- Attaching CDB zones to glTF nodes.
- Assigning switch nodes to a glTF node to represent damage states.
- Storing model attributes on a glTF node.
- Extending materials with CDB texture types: Detail texture, contaminants texture, and material map texture.

The full schema can be found here [https://github.com/sofwerx/cdb2-eng-report/blob/master/Experiment_Documents/glTF-CDB-extension-schema.zip].

The CDB glTF extension contains most of the required CDB-specific features. However there are some possible additions that could be made:

Coordinate reference system (CRS) Openflight can store the CRS of a model in its header and this capability is currently not possible in glTF. While not strictly required for CDB (A CRS of WGS 84 (EPSG 4326 and EPSG 4329) is mandated by the CDB standard), the 3D subgroup determined that this would be useful for interoperability. There is also interest in such an extension for One World Terrain (OWT). Therefore, specifying a CRS should likely be done as a stand-alone extension and not as part of the CDB glTF extension

Per face metadata The current OpenFlight specification allows for storing materials or relative priority per-face. However, in glTF faces are usually collapsed into meshes so there is no way to store information per polygon. With the current extension, the solution would be to group polygons with the same relative priority and material into a single mesh and store the information at the mesh level.

There is an OWT extension in progress to enable storing metadata at the feature level. This

extension [https://github.com/CesiumGS/glTF/pull/1] currently supports storing per-vertex metadata but that extension could possibly be expanded to support per-face metadata and used in a CDB datastore.

External References This is one of the larger challenges encountered during our investigations. In the past there was discussion about including an external reference extension in glTF but the Subgroup decided not to include such an extension. In past cases, separate index files were used to reference multiple glTF files from a common source. This is sufficient for some cases but does not help for the use case of sharing geometry between multiple models. For example sharing common roof clutter objects between different buildings

Extensions Roadmap

One of the challenges was deciding which CDB capabilities belong in `EXT_CDB` and which belong in standalone extensions. This is especially true when judged by an extension's utility to the wider glTF ecosystem. The 3D subgroup also investigated whether leveraging existing glTF extensions is possible. Finally, the 3D subgroup determined that some capabilities should go in the model indexing file rather than in the glTF. Our recommendations are below:

The EXT_CDB extension

- CDB-specific constructs such as points, zones, lights, light maps, and extended texture types.

Possible standalone extensions

- Coordinate Reference System - Shared interest with One World Terrain and possibly wider glTF ecosystem
- Switch nodes - Shared interest with One World Terrain moving models format
- External references - Some interest in wider glTF ecosystem, but many unresolved implementation details

Leverage existing extensions

- Articulations - AGI_articulations [https://github.com/KhronosGroup/glTF/tree/master/extensions/2.0/Vendor/AGI_articulations]
- Metadata - EXT_feature_metadata [https://github.com/CesiumGS/glTF/pull/1] - extension for storing metadata at different levels of granularity - per-mesh, per-face, per-vertex, and per-texel. This could be a good candidate for CDB model attributes and material map textures.

Model indexing file

- Model LODs
- Interior and exteriors
- Navigation and collision mesh

Adoption

The initial approach was to create a monolithic CDB glTF extension so that editors and viewers would only need to support a single extension to load CDB glTF models. However, the group

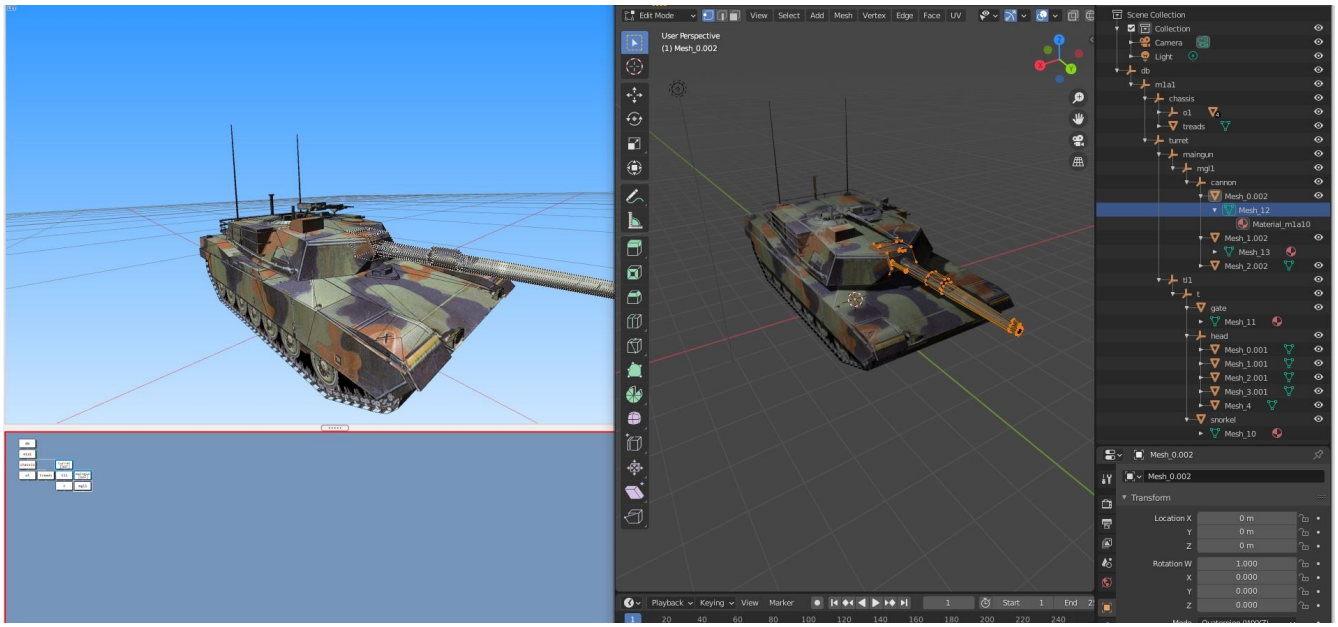
determined that there was a significant amount of overlap between CDB and OWT requirements and shifted towards a more granular approach to avoid multiple redundant extensions being developed. Generally glTF extensions that have seen wider adoption are both simple and useful to a wide variety of vendors. Suggested next steps are to collaborate more closely with OWT stakeholders and reach out to the wider glTF community for review. This review would be followed by practical experimentation in runtime engines. After OGC review and agreement, the extensions may be formalized as either ratified Khronos extensions (KHR prefix) or, more likely, Multi-Vendor extensions (EXT prefix), and be listed in the glTF Extension Registry [https://github.com/KhronosGroup/glTF/tree/master/extensions].

Another key topic for adoption is editor support. glTF has seen increasing adoption in 3D modelling software such as Blender, 3DS Max, and Maya but extensions are often lost during the import/export process. Workarounds have been proposed such as storing extension information outside of the glTF container (see The glTF Metadata File (GMDF) for Systems Tool Kit (STK) [https://github.com/AnalyticalGraphicsInc/gmdf]). Some capabilities like per-face feature assignment are not possible in standard modelling packages and would require plugins. The most likely path of adoption is native support for the extensions in simulation editors and plugins for more general purpose 3D modeling software.

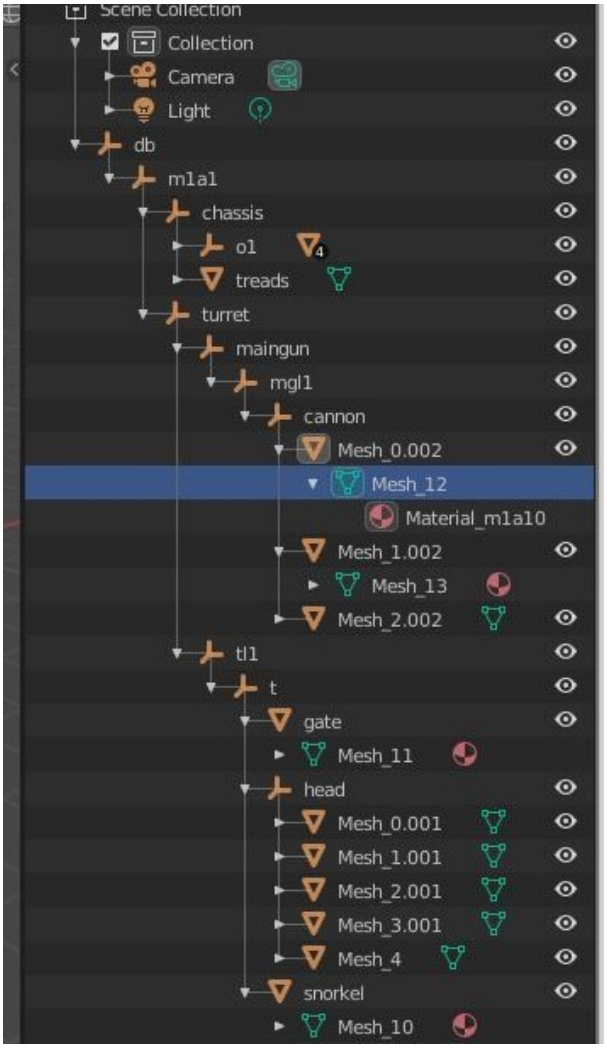
Sample model

test Model - tank [https://github.com/sofwerx/cdb2-eng-report/blob/master/Experiment_Documents/OpenFlight_to_glTF_testModel.zip] OGC Portal CDB tile in glTF [https://portal.ogc.org/index.php?m=projects&a=view&project_id=466&tab=2&artifact_id=95313]

11.2. TASK 3 - Prototype a OpenFlight to glTF converter supporting the new extension



[TankModel nodes OpenFlight] | *images/TankModel_nodes_OpenFlight.png*



Above are some images of a tank 3D model in both formats. This model was selected as benchmark as it contains a number OpenFlight specific constructs leveraged in the CDB standard. Converting this model will exercise {**exercized?**) many of the proposed glTF extensions.

OF/gltf size comparison metrics:

Small CDB size comparisons (898 GS models - without textures)

Comparing the size of the 300_GSModelGeometry and 303_GSModelDescriptor folders from the source CDB to the 999_ModelGeometry folder in the converted CDB {**models?**}:

Format	**Size**
Openflight	11.0 MB
Ascii gltf	9.16 MB
Binary glb	6.83 MB

Size comparison for a single very large model

Format	**Size**
Openflight	16.7 MB
Ascii gltf	9.27 MB

Binary glb	10.78 MB

In the CDB test case, the Ascii glTF file is slightly smaller than the Openflight file and the binary glb is almost twice as small. In the test case with a single large model, both the binary and ascii glTF provide a significant improvement in space savings. However, in this case the binary file is actually slightly larger than the ascii glTF. The reason for this is not entirely clear, but it seems to be an outlier based on teh results of the other tests.

11.3. TASK 4 - Define a new structure to store 3D models

Storing the 3D models in a CDB datastore can be either in files and folders or inside a GeoPackage. Where the prototype stopped short of packing {**Packing where**} the resulting glTF files, a number of elements need to be considered when this gets {**When what gets?**} investigated. Those are:

File based approach

1. Use zip: One challenge of using the CDB 1.X stadnrd is the large number of files that are created. Where many small files are not optimal on I/O, putting all datasets into one large file would lead to a large volume of data needing to be scanned to extract what is needed {**NOTE: This was investigated in the OGC Testbed 16 GeoPackage activity. Some interesting recommendations**}. A ZIP is a good format to allow quick access to the content index in order to decode only the required data. As such, a recommendation is to group each model data into a zip (all LODs, textures etc…) while keeping the xml (index file) separate. A quick access to the index file would allow identifying what is required to be loaded into each ZIP. This way, a client server could work with a scheme similar to the OGC getCapabilities to query the available data prior to accessing the actual payload.

GeoPackage based approach

<TBD>

11.4. TASK 5 - Define an index file for all 3D model components

The grouping of all components of a 3D model together requires an index file that indexes and references all those components. The list of components proposed include: Feature attribution (GGDM), reference to source model used, model exterior, Model interior, Collision volume, and nav mesh. Splitting those constructs into separate component follows the current logic in the CDB standard of splitting the dataset in order to quickly access only required data without having to load larger component(s) that include all {**All what?**} only to reject most of the content. This approach also serves the purpose of rapid update by allowing changing one element without the other (update nav mesh, or interior without changing the complete model) needing to be updated.

Some consideration should be made to ensure all existing GSModelDescriptor information is migrated into this file {**The index file?**} (texture mapping info being one).

Duplication of information There was a request to also store the model georeferencing inside this file {**The index file?**}. The challenge with this is the duplication of information such as CDB positions models based on the point feature. Changing the georeferencing in this file would not lead to a change in the model position. The CDB standard typically avoids duplication of information as

duplication creates confusion. In this same idea, one could argue that the model attribution (GGDB) should not be there either as this would also be duplication.

This file content and structure requires a review and alignment with the rest of the CDB metadata.

task: convert sample {**sample what?**} to object model and produce associated schema.

Sample XML model indexing file

```
<?xml version="1.0" encoding="utf-8"?>
<!-- "Model" node includes encoding statement. Should this info be pushed down,
     maybe in the geometry node? Do we allow multiple encodings all referenced
     in the same indxing file?
-->
<Model encoding="OpenFlight" multi-instance="false">
    <!-- "Name" has to be unique in a tile. Do we need this info or we rely on the
filename
         or geopackage entry name.
    -->
    <Name>cdb_swa_gs_hotel_03</name>
    <Identification>
        <GGDM>
            <Code>AL015 </Code>
            <Subcode>0 </Subcode>
        </GGDM>
    </Identification>

    <!-- model_complete_source - Do we allow uses/users to store this in CDB? Where?
Any format?
         This model would typically include all LODs and possibly constructs not
supported
         by CDB. This would likely also require the storing of the source textures. As
such,
         a zip is likely a better container. Still, inclusion of this will increase
CDB size
    -->
    <model_complete_source file="My_hotel_model_full_Detail.obj" encoding="obj"/>

    <!-- model_exterior - Following current CDB, exterior (shell) of a model should be
seperated
         from interiror and other mesh representation for load/stream efficency.
    -->
    <model_exterior>
        <LODherarchy coarsestCDBLOD="1" finestCDBLOD="5">
            <RootModel>  <!-- root model points to all LODs -->
                <geometry file="cdb_swa_gs_hotel_03_ROOT.flt"/>
                <Metadata>myFile.xml</Metadata>
            </RootModel>
            <LevelOfDetail LOD="2">
                <geometry>
                    <file>cdb_swa_gs_hotel_03_L02.flt</file>
```

```
                </geometry>

<Metadata>N12E044_D303_S001_T001_L02_U1_R1_AL015_000_cdb_swa_gs_hotel_03.xml</Metadata
>
                <!-- need to add provenance to metadata -->
            </LevelOfDetail>
            <LevelOfDetail LOD="5">
                <geometry>
                    <file>cdb_swa_gs_hotel_03_L05.flt</file>
                </geometry>

<Metadata>N12E044_D303_S001_T001_L05_U1_R1_AL015_000_cdb_swa_gs_hotel_03.xml</Metadata
>
                <!-- need to add provenance to metadata -->
            </LevelOfDetail>
        </LODherarchy>
    </model_exterior>
    <!-- model_interior - The linkage between exteriror and interior geometry
currently relies on XREF in
         OpenFlight in order to position/connect interior with exterior. (requirement
for glTF)
         Alternatively, model with the interior could be a higher LOD of an object.
This is not precluded
         by the standard (as long a geometry comply to CDB geometry LOD rules).
    -->
    <model_interior>
        <LODherarchy coarsestCDBLOD="5" finestCDBLOD="5">
            <RootModel>  <!-- root model points to all LODs -->
                <geometry file="cdb_swa_gs_hotel_interior.flt"/>
                <Metadata>myFile.xml</Metadata>
            </RootModel>
            <LevelOfDetail LOD="5">
                <geometry>
                    <file>cdb_swa_gs_hotel_interior_L05.flt</file>
                </geometry>

<Metadata>N12E044_D303_S001_T001_L01_U1_R1_AL015_000_cdb_swa_gs_hotel_03.xml</Metadata
>
                <!-- need to add provenance to metadata -->
            </LevelOfDetail>
        </LODherarchy>
    </model_interior>
    <collision_volume/>
    <navigation_mesh/>
</Model>
```

11.5. Investigation 2 - Merging all geometry in a tile

This investigation was not done by the 3D group but the tiling group did experiement. A few different ways to encode are possible, each with their own implications.

1. A mesh of the terrain geometry, esentially converting the CDB elevation and imagery datasets into one dataset. Where a quick implementation is possible one must consider all sub-datasets combinaisons (bathymetry for elevation and all imagery variants) and their implications in teh CDB data model.
2. A mesh of only 3D models (not terrain) per tile. This has the benefit of improving rendering efficency as one tile. Some of the challenges to consider here are the ability to clamp individual models to terrain elevation while preserving the geometry grouping benefits. Also, the attribution on a per model basis currently contained in the vector would need to either move inside the mesh (to identify mesh segment per feature) or have a wait for the vector to reference part of the tile mesh.

Generally, this type of solution has significant impact on the CDB data model: . The mesh solution merges many CDB datasets into one. The elevation and 3D features along with the imagery are now all merged into one dataset. . The attribution to objects on the terrain is currently done in the vector, with a reference to the geometry. In a mesh case, the linkage will likely need to be revered and have the geometry link to the attribution.

Chapter 12. CDB X Metadata and Integration

This section discusses the role of metadata in CDB-X.

12.1. Summary of history of CDB-X metadata discussions.

The earlier 'industry' CDB versions and the OGC CDB 1.x versions of CDB contain a 'metadata' directory. Despite the directory name, this is a collection of .XSD (XML Schema Definition) and associated .XML (eXtensible Markup Language) files that are metadata, controlled vocabularies, and enumerations for elements of the CDB standard.

NOTE

More specifically, the majority of what is referred to as metadata in the CDB standard is not metadata! The majority of the schemas, metadata references, and related requirements are actually controlled vocabularies (code-lists). Lights, materials, and so on are controlled vocabularies and not metadata. Some elements of the standard, such as “version” are referred to as metadata and could be classed as metadata.

In the OGC CDB Github "Schema" repo, the contents of the root level are:

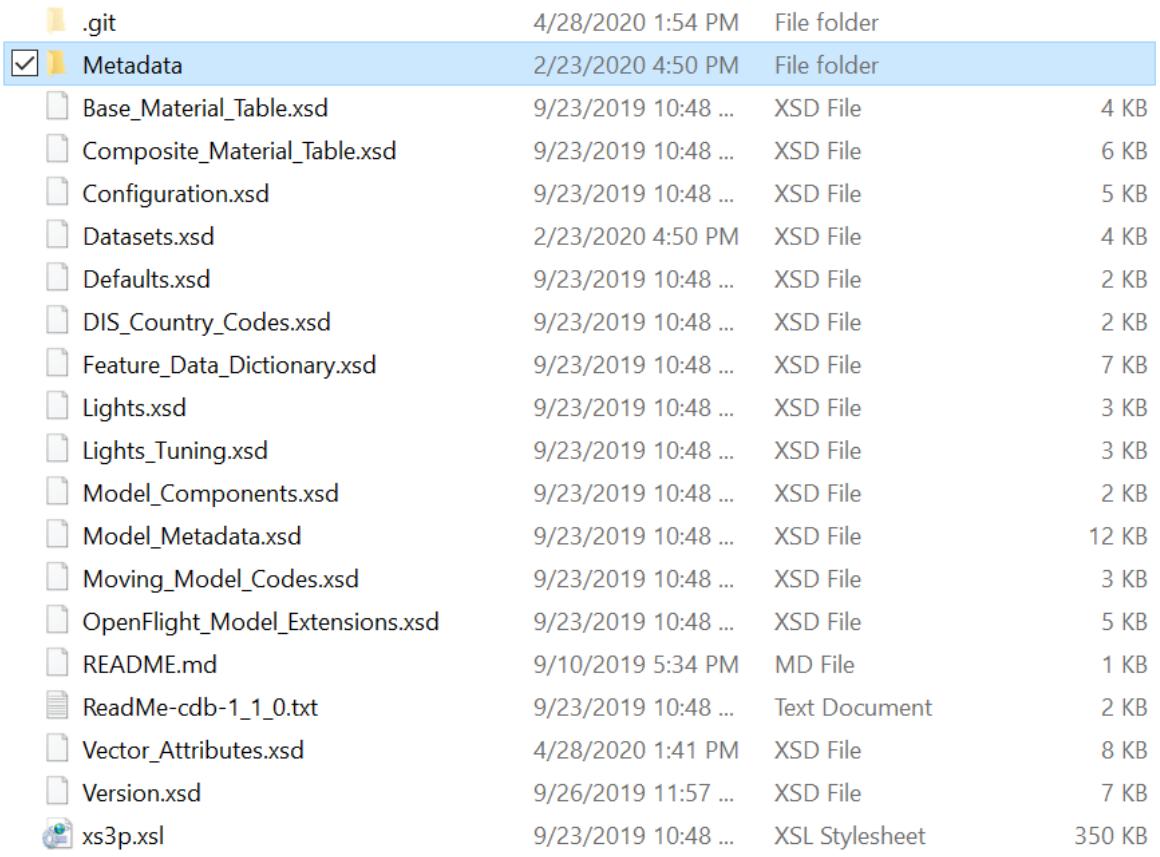


Figure Mtd_Ph3 - 24. OGC CDB Schema Github Repo Root level files.

and the contents of the Metadata subdirectory are:

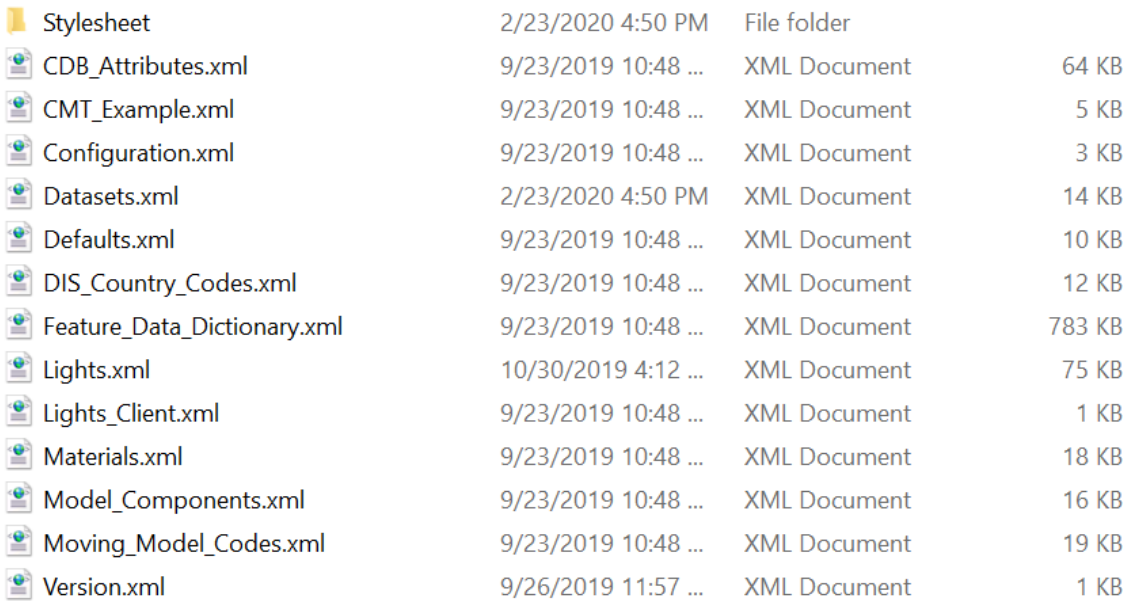


Figure Mtd_Ph3 - 25. OGC CDB Schema Github Repo Metadata sub-directory files.

OGC CDB V1.1 is a minor revision of OGC CDB Standard that provides guidance and recommendations for optional global and local metadata in a CDB datastore. The description of the new geospatial metadata provisions is in Section 5.1 of the CDB Standard Volume 1 Core document.

During the CDB 1.1 revision discussions, the SWG evaluated metadata elements as specified in a number of international standards as well as DoD metadata specifications. The following metadata standards, widely referenced and used in the geospatial community, were considered as input to the recommendations:

- Dublin Core
- DCAT
- INSPIRE/19115
- DDMS 1 (2003)
- DDMS 5 (2012) M&S Profile
- NOAA 19115 Profile
- FGDC
- GeoDCAT-AP
- US Federal Open Data Metadata schema (see annex A)

Evaluation of these metadata standards led to defining a limited set of metadata specified as guidance in the CDB 1.1 Standard. For ease of reference, sections 5.1.10 thru 5.1.12 from Volume 1 of OGC CDB V1.2 are repeated here:

12.1.1. Geospatial Metadata – Guidance

These are optional metadata files. This file is not included with the CDB distribution schema package.

Most metadata standards specify dozens of possible elements, such as author, that can be specified in a metadata encoding. This is why in many communities there are profiles that are applicable to the information sharing and discovery requirements of that community. For example, there are numerous profiles of ISO 19115:2013 Geographic information – Metadata. These include the INSPIRE, Defence NSG Geospatial Core metadata, and FGDC profiles. As such, the CDB standard does not specify mandatory and/or optional metadata elements. Instead, a suggested set of minimal metadata elements are provided. The two lists – one for global and one for local – are based on an evaluation of mandatory elements in eight widely implemented metadata standards that are used in the geospatial and simulation communities. The one requirement is that all local metadata in a CDB data store provides the same mandatory elements as defined in the metadata standard specified in the Version metadata.

These following two sub-clauses recommend the metadata elements for global and local metadata. The use of F.1 refers to Table F.1 in ISO 19115-1:2014. Each element is identified by a general string followed by two element names The first name is the DCAT name followed by the ISO 19115:2014 element name.

Suggested Global Geospatial Metadata Elements

Resource Identifier (dct:identifier, MD_Metadata.metadataIdentifier): A unique identifier for the entire CDB data store instance. This identifier is persistent and is considered global metadata. For example, this could be a Digital Object Identifier (DOI). The **DOI** system provides a framework for persistent identification of electronic resources management of intellectual content, managing metadata, linking customers with content suppliers, facilitating electronic commerce and enable automated management of media.

Resource Title (dct:title [http://purl.org/dc/terms/title], CI_Citation.title): Title by which the resource is known (Table F.1). For global metadata for a CDB data store, this would be a name given to the entire data store. For example, this could be “Yemen demonstration CDB data store.”

Resource point of contact (dcat:contactPoint, (MD_Metadata.contact/CI_ResponsibleParty): Name of the person, position, or organization responsible for the resource. (Table F.1). This is a text string. An example of a resource point of contact could be “Flight Safety” or “CAE.”

Resource reference date (dct:issued, CI_Citation.date): A date which is used to help identify the resource. (Table F.1). For global metadata, this is the date that the CDB data store was created or issued.

Resource Language (dct:language, PT_Locale): The language and character set used in the resource (if a language is used). (Table F.1) NOTE: We should recommend use of ISO 639-2 . For example, for English, the code would be “ENG.”

Geographic Location (dct:spatial, EX_GeographicBoundingBox): Geographic description or coordinates (latitude/longitude) which describes the location of the resource. Note: I think for the CDB standard that the definition should be narrowed to the bounding box of the contents of the data store. (Table F.1). We should also follow guidance from OGC OWS Common. See also 19115 annex B.3.1.2 Geographic extent information.

Resource abstract (dct:description [1], MD_DataIdentification.abstract): A brief description of the content of the resource (Table F.1).

Metadata date stamp (dct:issued, MD_Metadata.dateInfo): Reference date(s) for the metadata, especially creation. (Table F.1). Note: Date gives values for year, month and day. Character encoding of a date is a string which shall follow the format for date specified by ISO 8601. This class is documented in full in ISO/TS 19103.

Temporal Extent information for the dataset (dct:temporal, EX_TemporalExtent): The temporal extent of the resource. For a CDB data store, this would be the temporal range of when the data store was initially created to the point where the most recent content was created.

Constraints on resource access and use (dct:accessRights, MD_SecurityConstraints): Security restrictions on the access and use of the resource. These would be constraints for an entire CDB data store. This could be information necessary to generate an EDH compliant encoding.

Constraints on resource access and use (dct:license, MD_LegalConstraints): A sub-class of all access constraints. These legal constraints include copyright, patent, patent pending, trademark, license, Intellectual Property Rights, restricted, and other. At the global level, these are legal constraints applicable to an entire CDB data store.

Suggested Local Geospatial Metadata Elements

Local Geospatial metadata can be stored in a number of different folder locations based on the data resource (data set) for which the metadata is associated. For instance, metadata for vector data will be stored at the LoD/tile level. Metadata for a moving model would be stored in the same folder using the same path name as the actual model definition. See Clause 5.1.2 above for examples.

+ While the same metadata elements are recommended for both global and local geospatial metadata, there are some differences that should be considered.

+ *Metadata Reference Information* (dct:identifier, MD_Metadata.metadataIdentifier) This is a unique identifier for the dataset. In CDB, this could be the pathname to the dataset or the tile. These pathnames are unique. Using such identifiers would facilitate development of a RESTful API for discovery and access of CDB resources.

+ *Resource Title* (dct:title [http://purl.org/dc/terms/title], CI_Citation.title): Title by which the data set is known (Table F.1). For local metadata, this could be a name given to a layer or model in the data store. In a CDB data store, at the dataset or tile level this would be a name given to the resource, such as “county soils.”

+ *Resource point of contact*: Name of the person, position, or organization responsible for the resource. This is a text string. An example of a resource point of contact for the content for a given layer and tile could be “Ordnance Survey.”

+ *Resource reference date* (dct:issued , CI_Citation.date): A date which is used to help identify the resource. For local metadata, this could the date that the tile content was created in the CDB data store or the date a moving model was added to the data store

+ *Spatial Resolution Information* (No equivalent, MD_Identification.spatialResolution): The nominal scale and/or spatial resolution of the resource. This description can include LoD information. Note: This is not precision! Precision is more about the number of decimal places and not the accuracy of the resource.

Where are local metadata files stored?

Typically, local metadata files will be stored with the physical data. For GTModel geotypical data sets, the metadata file would be stored along with the model XML file. If the model is stored in multiple LoDs, the metadata would also be stored at each LoD. For tiled vector data, the local metadata would be stored with the vector files at the tile level.

12.1.2. Recommendations

The following are the recommendations for metadata and provenance in CDB-X. Please note that all Sprint participants agreed that metadata including provenance is a critical requirement for the CDB-X Standard. They also agreed that some elements should be mandatory.

1. Metadata and provenance content should be self-describing.
2. Keep the core set of mandatory metadata elements limited and simple. Collecting and maintaining metadata can be costly – unless workflows are designed to capture metadata as part of the production process.
3. Define an extensible CDB metadata model that allows for easily incorporating additional metadata elements for specific data types, domains or applications. A good example would be the metadata required to make SWIR and NIR in a CDB data store useful by discovery, access, processing, and visualization services for those data types.
4. Discuss and agree on element names for the mandatory elements. This is because each metadata standard names elements differently. This also suggests that a metadata element crosswalk document may be required. The beginnings of such a document were developed as part of the CDB 1.1 revision work.
5. Every CDB dataset should have its own metadata that describes the content of that specific dataset. This will allow for much more flexible extensibility of new data types and enhances the value of existing datasets and enhances discoverability.
6. Consider whether the GeoPackage Metadata extension is robust and flexible enough to meet CDB-X requirements.

12.1.3. Future vision of role of metadata in CDB-X

The following diagram provides a suggestion of where CDB-X metadata could exist.

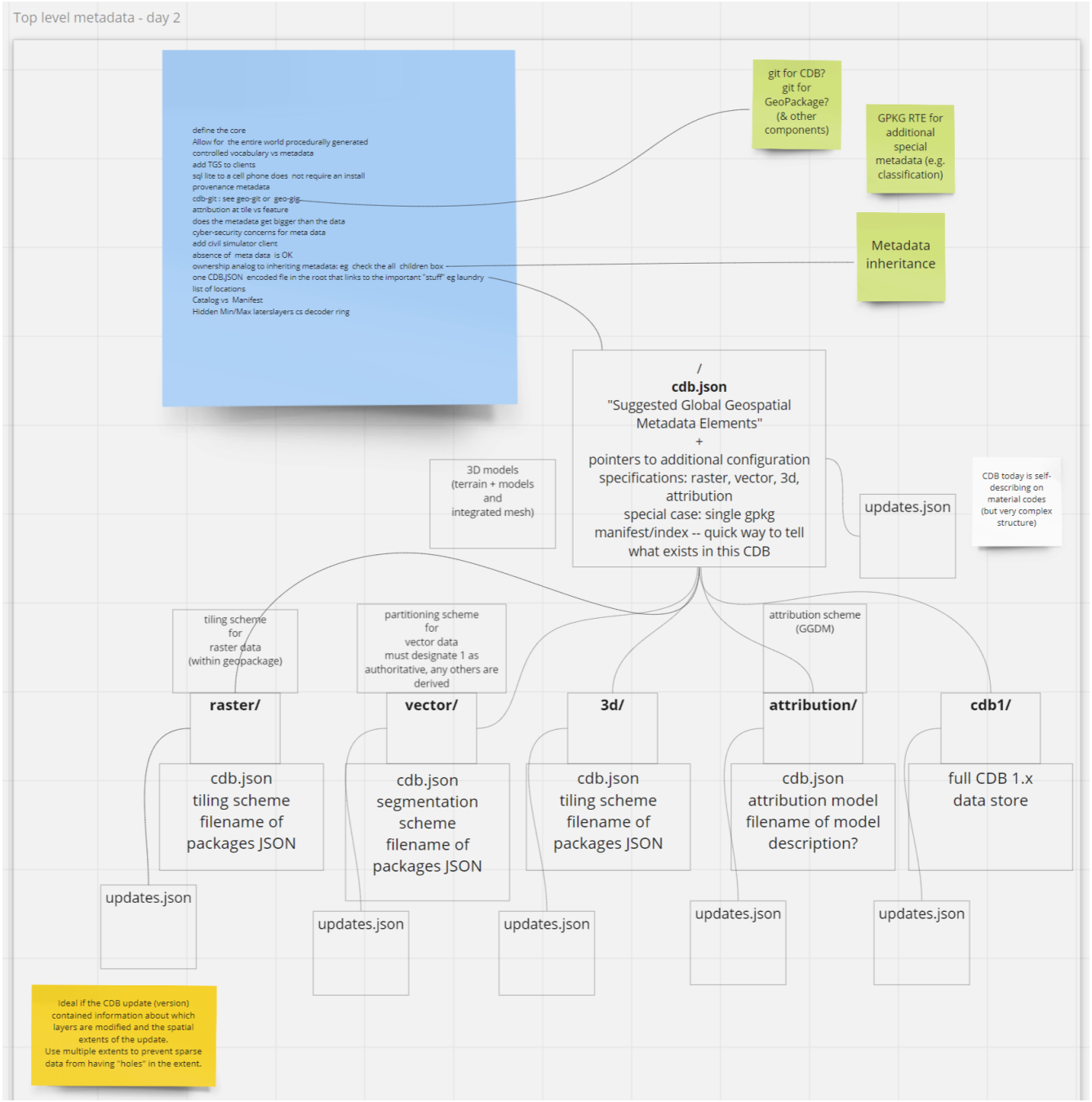


Figure Mtd_Ph3 - 26. Metadata Day 5 Whiteboard.

[1] DCAT does not have a concept “abstract”. Use description instead.

Chapter 13. Tiling and packaging of data layers

This chapter covers:

- The evaluation of a specific global tiling scheme for CDB X,
- The configurable grouping of Levels of Detail to balance file size versus file count,
- An approach to packaging one or more data layers in GeoPackages (including coverage, vector and 3D data),
- Current and proposed GeoPackage extensions for storing this data, and
- The results of related prototyping activities.

The tiling sub-group was initially assigned specifically to explore tiled coverage data However, based on initial work, some experiments evolved beyond the initial testing scope. The additional experimentation applied the same tiling approach to vector data and 3D models and used existing or new extensions to store this tiled data in GeoPackages.

13.1. OGC Standards Relevant to Tiling and Packaging

A key particpant decision was what container technology would be used for the majority of the experiments in which tiled structures are required. For the purposes of experimentation coupled with a community and sponsor reqwuirements, OGC GeoPackage structured containers were determined to be the primary storage format for derived vector data layers, coverages, and other data types as identified (TBD). The two OGC GeoPackage Standards of relevance are OGC GeoPackage version 1.1 [https://portal.opengeospatial.org/files/12-128r15] and OGC GeoPackage Extension for Tiled Gridded Coverage Data [http://docs.opengeospatial.org/is/17-066r1/17-066r1.html].

Further, based on the discussions and results of the experimentation, the Tiling Subgroup participants believe the following recommendation.

Recommendation: Any tiling schemes specified in a CDB X data store (repository) SHALL be based on and consistent with the:

- OGC Core Tiling Conceptual and Logical Models for 2D Euclidean Space [https://portal.ogc.org/files/?artifact_id=92962&version=1] (19-014r3)
- OGC Two Dimensional Tile Matrix Set Standard [https://www.ogc.org/standards/tms] (17-083r2)

Additional information is now provided on these key OGC standards that are highly relevant to the development and approval of CDB 2.0.

13.1.1. OGC Core Tiling Conceptual and Logical Models for 2D Euclidean Space

This OGC Abstract Specification consists of two Parts: A General Tiling Conceptual Model and, based on the Conceptual Model, a Logical Model for the Tessellation (Tiling) of 2D Euclidean Space. Tiling

of 2D Euclidean space is the most commonly known approach to partitioning space in traditional geospatial technology. However, there are also common elements and/or semantics for any approach to partitioning space in any dimension. The logical model in this document defines a set common required elements and then follows with more specific requirements for the two dimensional case.

Part 1 of the Abstract Specification describes a general tiling conceptual model. The conceptual model is applicable to any dimension. The conceptual model makes no assumptions regarding content, use cases, implementation scenarios, or how the space is to be tessellated (tiled). The conceptual model is abstract and cannot be implemented as is.

Part 2 of the Tiling Abstract Specification defines a detailed logical model for the tessellation of 2D Euclidean Space. One or more logical models are required to provide the requirements and structure necessary for implementation. Therefore, in addition to the conceptual model, this Abstract Specification also specifies a core logical model for the 2D planar (Euclidean) use case.

13.1.2. OGC Two Dimensional Tile Matrix Set Standard

The OGC Tile Matrix Set standard defines the rules and requirements for a tile matrix set as a way to index space based on a set of regular grids defining a domain (tile matrix) for a limited list of scales in a Coordinate Reference System (CRS) as defined in [OGC 08-015r2] Abstract Specification Topic 2: Spatial Referencing by Coordinates. Each tile matrix is divided into regular tiles. In a tile matrix set, a tile can be univocally identified by a tile column a tile row and a tile matrix identifier. This document presents a data structure defining the properties of the tile matrix set in both UML diagrams and in tabular form. This document also presents a data structure to define a subset of a tile matrix set called tile matrix set limits. XML and JSON encodings are suggested both for tile matrix sets and tile matrix set limits. Finally, the document offers practical examples of tile matrix sets both for common global projections and for specific regions.

13.2. Design Objectives for CDB X Tiling, LoDs and Layers

The following are the design principles for a CDB X tiling, levels of detail, and layering. These design principals were discussed and documented during Phase 1 discussions as well as during the experimentation. Again, for the purposes of expediting experimentation, GeoPackage was the choice for a standard container.

- Use metadata for whole datasets that describe how data layers are regrouped (LOD grouping and data layer grouping).
- Provide for fewer top level tiles than in OGC CDB 1.x.
- Store imagery in a container according to the tiling scheme specified below.
- Store coverage data in the container according to the tiling scheme specified below (need anchor). Suggested coverage types based on current CDB standard and user specified requirements: (Ellipsoidal body surface): Elevation models, visual and non-visual (multi-spectral) imagery, terrain light maps (emissive at night), and raster materials (for non-visual sensors). Should be extensible to support other types.

- Coordinate Reference System is WGS 84 with epoch encoding (same as current CDB Standard except for epoch).
- Need to keep the various encodings that CDB 1.x already has:
  - Elevation compression encodings (use of scaled integers vs floating point for smaller data sizes).
  - Elevation grids that are adjustable (better terrain fidelity at lower LODs/zoom levels)
  - Raster Material encodings using multiple coverage layers.
  - Imagery Compression (Imagery is typically the largest layer in CDB by disk storage).
- Must enable a "relatively" easy migration path from the CDB 1.1/1.2 tiling/LoD structure into the CDB X structure.

13.3. Proposed Tiling Logical Model for CDB X

The following figure, based on the Tiling Abstract Specification Logical Model and the CDB 1.x Conceptual Model, shows the properties by class (concept) and the relationships between the classes.

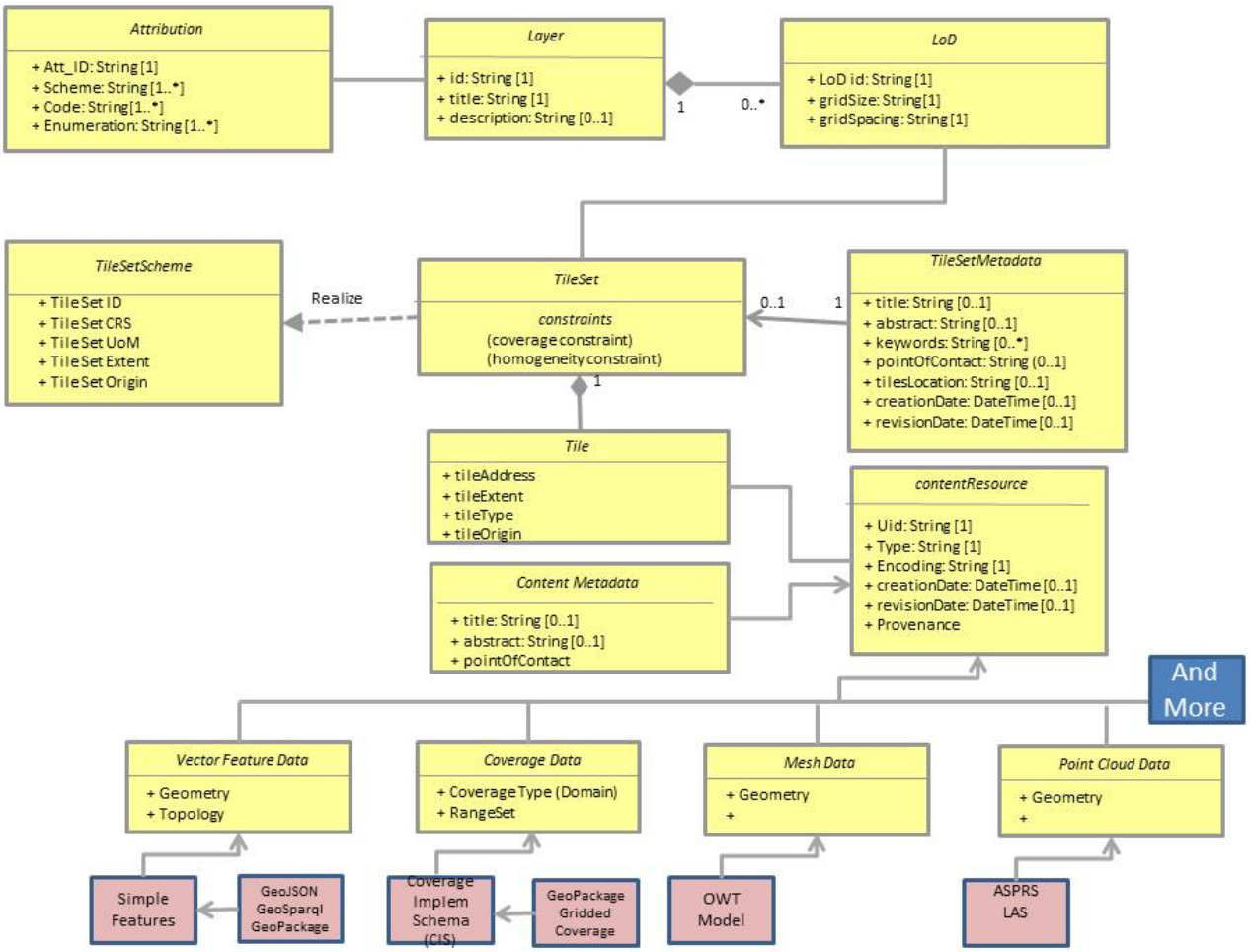


Figure Mtd_Ph3 - 27. Logical Model for partitioning based on tiles in CDB X.

13.4. Proposed CDB X Tiling Scheme

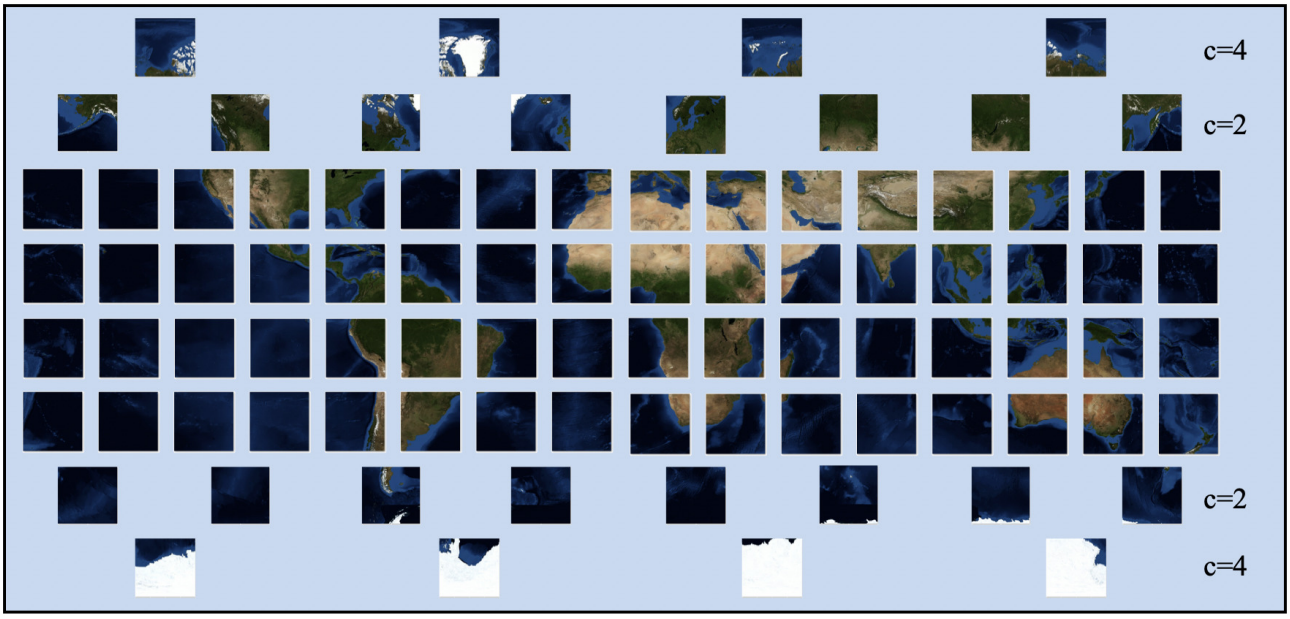


Figure Mtd_Ph3 - 28. Level 2 of the GNOSIS Global Grid proposed for CDB X.

The proposed tiling scheme would be based on the GNOSISGlobalGrid, where the tiling starts with a 2x4 grid of tiles with each tile 90 degrees on a side that covers the whole Earth. This is zoom level 0 (or tile matrix identifier "0"). The experiments performed during the sprint used the tile size of 256 x 256 defined by the GNOSIS Global Grid for all zoom levels / level of details. A variation of the Tile Matrix Set definition using a different tile size instead, e.g. 512 x 512 could also be experimented it. At each successive zoom level, each lower level tile is split into four new tiles at the next zoom level The exception that any tile that touches either the North or South Pole is not split in the longitude direction. This subdivision can continue until the zoom level is high enough to accommodate the highest resolution data that is to be stored within the CDB X.

References for the GNOSISGlobalGrid tiling scheme:

- The 2D TMS Standard Annex H.2 [http://docs.opengeospatial.org/is/17-083r2/17-083r2.html#106]
- The 2D TMS JSON description [http://schemas.opengis.net/tms/1.0/json/examples/GNOSISGlobalGrid.json]
- Testbed 13 - Vector Tiles Engineering Report [http://docs.opengeospatial.org/per/17-041.html#_global_gnosis_tiling_scheme_adapted_to_polar_regions]
- OGC Standard Tracker - Global WGS84 tiling scheme [http://ogc.standardstracker.org/show_request.cgi?id=520] adapted to polar regions (quad tree except for always having 4 tiles at the poles)
- OGC Ideas Repository - Global tiling grid approximating equal-area [https://github.com/opengeospatial/ideas/issues/59] while maintaining a simple latitude/longitude aligned rectangular tile layout (not quite a DGGS).
- Vector Tiles Pilot - Phase 2 Summary Engineering Report [https://docs.ogc.org/per/19-088r2.html]

Figure Mtd_Ph3 - 29. Proposed Tiling Scheme for CDB X.

13.5. Findings from experiments

The following graph compares the CDB 1.x zones with the Gnosis grid and shows how the GNOSIS algorithm helps to keep the typical tile closer to a “square” than CDB’s zones.

Figure Mtd_Ph3 - 30. CDB 1.x to GNOSIS Comparison - Ratio.

NOTE A summary of all of the results from the experiments can be found in Annex B.

13.5.1. Tiling Changes From OGC CDB 1.x

Benefits from changing to the proposed tiling scheme:

- OGC CDB 1.x has 41,220 top level tiles. This requires opening huge numbers of files to visualize the Earth at global scales. The proposed CDB X tiling would use 8 tiles to cover the Earth at the coarsest level of detail.
- The concept of "zones" in OGC CDB 1.x are still present in the tiling scheme, but algorithmically derived rather than at fixed latitudes. New zones are introduced at higher levels of detail, keeping the tiles near the poles closer to an ideal square shape than in OGC CDB 1.x
- The ratio of the longitude size of tiles is always 1:1, 2:1, or the inverse 1:2, where OGC CDB 1.x has several ratios that must be supported, such as 2:1, 3:2 and 4:3.

Drawbacks to changing from the CDB 1.x approach to the proposed tiling scheme include the following. NOTE: These issues would require some level of rework for existing CDB applications to be compatible with the new tiling scheme.

- The new tiling scheme is incompatible with OGC CDB 1.x, as there is no alignment between the tile areas and the LOD or zoom levels. To convert data between these two tiling schemes would require merging and splitting of raster data tiles, while changing their resolution, and would require reprocessing all the coverage data (like imagery).
- Some CDB applications might have more trouble with tiles that are not based on integer latitude

and longitude boundaries.

- Some CDB applications might have an issue with a format where there are not a fixed number of "zones" (using the OGC CDB 1.x term) or different grid cell sizes used. This is because the GNOSIS grid introduces a new "zone" closer to the poles at each successive zoom level to help preserve a grid cell that is closer to an ideal square real world size.
- Tile Matrix Set naming (numbering) is different from OGC CDB 1.x, with the numbering starting from the top left corner of a set of tiles in CDB X versus starting from the bottom left corner of a set of tiles in OGC CDB 1.x.

13.6. Proposed CDB X Data Container

The coverage, vector and 3D models data could be stored using GeoPackage [http://www.geopackage.org/spec/] containers. Imagery and coverage data would always be tiled. Vector data could optionally be stored in tiles using extensions for tiled vector data being standardized in the OGC. 3D models could either be stored in a single table referenced by placement points, or as batched tiled 3D models.

13.7. Proposed Levels of Detail Grouping

There are two proposed ways to group data within a series of GeoPackage containers. Within these choices is a tradeoff between the format simplicity of working with a single container, and access latency due to larger file and table sizes.

1. For users at the edge and smaller areas, the Subgroup particpants `recommend` that all the CDB X coverage layers be present within a single GeoPackage container. This approach could be result from an export from a larger CDB X dataset, where only the resolutions and the boundaries of the data needed are stored and delivered to an end user. Users at the edge are such individuals as field operatives using mobile devices with limited battery, memory, storage and processing capabilities as well as Delayed/Disconnected, Intermittently-connected or Limited bandwidth (DDIL) environments.
2. For Modeling and Simulation uses, as well as data repository cases, the Subgroup participants `recommend` that a series of GeoPackage containers be used to store CDB X coverage layers. This involves storing a region of the world at a set of zoom levels, or levels of detail, within a single GeoPackage, with a specified file and directory naming scheme. This approach would allow for faster access to data at specified areas and zoom levels. This approach would also lend itself to concurrent access and editing for data repository maintainers.
    - The particpants propose that a configurable grouping value be used to specify how many zoom levels or levels of detail are put into a single GeoPackage. This would be a tradeoff between the number of GeoPackages created and the file and table sizes within each GeoPackage.
    - Multiple grouping numbers were experimented with. Five (5) was found to be a good number for packaging together all data layers of the San Diego CDB. For these experiments, a number of datasets were used, including CDB 1.x data from both the Presagis Sample CDB of Camp Pendleton, as well as the San Diego sample CDB provided by CAE. Additional data from NASA Visible Earth Blue Marble and USGS GTOPO30 were used to test applying the

tiling to global datasets. This grouping number might need to be adjusted for different types of data layers, or different data sets as well.

- The proposed naming of each GeoPackage file is based on the layer name and the coarsest, lowest level tile included within the grouping inside the GeoPackage. That tile's level, row (from the top) and column within the tile matrix set make up the filename, along with the level of the finest or highest level tile that can be placed into this GeoPackage. For example: **Coverages_L4_R16_C12_L6.gpkg**
- The proposed directory naming creates a directory tree that limits the number of GeoPackage files that could exist within a single directory. Each GeoPackage would exist within a set of directory names that represents each coarser or lower zoom level GeoPackage that encompases the smaller higher resolution area. For example: The file, **Coverages_L4_R16_C12_L6.gpkg**, would exist in the directory named **Coverages\L0_R1_C0\L1_R2_C1**
  - The file and directory naming needs to be easy to compute algorithmically or exist within a catalog, without having to search a data repository to discover arbitrarily named files.

To facilitate extraction from a large (or even worldwide) dataset and easily merge these extracted sub-datasets as well as the ability to augment these base datasets with additional more detailed insets, without repackaging the entire dataset, the following solutions are proposed:

- The publisher of a large repository of data could pre-establish fixed maximum LODs for specific data layers (e.g. imagery, elevation, 3D Models).
- When inset data is added to a dataset, these could go beyond the regular maximum LOD specified for the 'packaging'. As such these inset tiles are always grouped together with the tiles of the regular maximum LOD.

`Recommendation`: Define this capability for splitting GeoPackages based on a specific tiling scheme outside of the CDB X standard so that this split content can be used by itself.

NOTE A mechanism to split data and group (or not) layers over multiple GeoPackages by tiles / LOD grouping (or have it all as a single GeoPackage), as well as a JSON schema providing the necessary information to access that data (e.g. packages, data layers, grouping, maxLOD, selected TileMatrixSet), should be a modular standard defined outside of CDB X, so that it could be used for other large datasets not necessarily conforming to the CDB X standard (e.g. the OrdnanceSurvey Master Map, OpenStreetMap Planet OSM, etc.).

13.8. Proposed `cdb.json` index of packages and data layers

The Tiling Subgroup developed a simple schema for describing the data layers provided in a CDB X datastore, which CDB component selector they correspond to, and how they are packaged in one or more *packages*. These *packages* can themselves either be stored in a single GeoPackage (zero or *null* LOD grouping), or separated in multiple GeoPackage based on tiles grouped in multiple LODs (non-zero LOD grouping). This index allows the flexibility to use the best suited configuration for a given

dataset or use case At the same time it is very simple for a client to parse and access the data in a deterministic manner using any of these configurations.

When a single GeoPackage is used, potentially this `cdb.json` could be included inside as metadata (using the GeoPackage metadata extension) to make this GeoPackage a single file, very portable compliant CDB X. The following `cdb.json` index files demonstrate the three main configuration possibilities. Additional examples are also provided along with the experiments results.

Example 1: Two data layers (with a maximum LOD of 16 and 18) packaged as a single GeoPackage ("groupLOD" : 0 — all LODs grouped together). This would be a single file named `SampleCDBX.gpkg`.

```
{
   "packages" : [
      {
         "name" : "SampleCDBX",
         "tms" : "GNOSISGlobalGrid",
         "groupLOD" : 0,
         "maxLOD" : 18,
         "layers" : [
            {
               "name" : "Elevation",
               "cs1" : 1,
               "maxLOD" : 16
            },
            {
               "name" : "Imagery",
               "cs1" : 4,
               "maxLOD" : 18
            },
         ]
      }
   ]
}
```

Example 2: Two data layers (with a maximum LOD of 16 and 18) packaged as a single package, grouped in GeoPackages covering tiles across 5 LODs. The one package would be organized in a folder named `SampleCDBX`, and the GeoPackages files will be grouped by 5 counting down from the maximum LOD 18, e.g. `SampleCDBX/L0_R0_C0/L4_R10_C11/L9_R324_C356/SampleCDBX_L14_R10376_C11415_L18.gpkg`.

```
{
   "packages" : [
      {
         "name" : "SampleCDBX",
         "tms" : "GNOSISGlobalGrid",
         "groupLOD" : 5,
         "maxLOD" : 18,
         "layers" : [
            {
               "name" : "Elevation",
               "cs1" : 1,
               "maxLOD" : 16
            },
            {
               "name" : "Imagery",
               "cs1" : 4,
               "maxLOD" : 18
            }
         ]
      }
   ]
}
```

Example 3: Three data layers (with a maximum LOD of 16, 18 and 13) packaged as three separate packages. Elevation is grouped in GeoPackages covering tiles across 6 LODs, Imagery is grouped in tiles across 7 LODs while Buildings are stored grouped in a single GeoPackage. The Elevation package would be organized in a folder named `Elevation`, and the GeoPackages files would be grouped by 5 counting down from the maximum LOD 16, e.g. `Elevation/L0_R0_C0/L5_R20_C22/Elevation_L11_R1302_C1427_L16.gpkg`. The Imagery package would be organized in a folder named `Imagery`, and the GeoPackages files would be grouped by 7 counting down from the maximum LOD 18, e.g. `Imagery/L0_R0_C0/L5_R20_C22/Imagery_L12_R2605_C2855_L18.gpkg`. The Buildings would be stored in a single file named `Buildings.gpkg`.

```
{
   "packages" : [
      {
         "name" : "Elevation",
         "tms" : "GNOSISGlobalGrid",
         "groupLOD" : 6,
         "maxLOD" : 16,
         "layers" : [
            {
               "name" : "Elevation",
               "cs1" : 1,
               "maxLOD" : 16
            }
         ]
      },
      {
         "name" : "Imagery",
         "tms" : "GNOSISGlobalGrid",
         "groupLOD" : 7,
         "maxLOD" : 18,
         "layers" : [
            {
               "name" : "Imagery",
               "cs1" : 4,
               "maxLOD" : 18
            }
         ]
      },
      {
         "name" : "Buildings",
         "tms" : "GNOSISGlobalGrid",
         "groupLOD" : 0,
         "maxLOD" : 13,
         "layers" : [
            {
               "name" : "Buildings",
               "cs1" : 100,
               "cs2" : 1,
               "subComponent" : 1,
               "maxLOD" : 13
            }
         ]
      }
   ]
}
```

Recommendation: Consider also defining this description of the packages and LOD grouping outside of the CDB X standard so that description can be used elsewhere as well.

13.9. Backwards Compatibility with OGC CDB 1.x

The current OGC CDB tiling scheme can be described as a Tile Matrix Set (TMS) that encodes the CDB fixed zones and the larger tile dimensions, as seen in this example description [https://maps.ecere.com/ogcapi/tileMatrixSets/CDBGlobalGrid] (JSON encoding [https://maps.ecere.com/ogcapi/tileMatrixSets/CDBGlobalGrid?f=json]).

Using this TMS, a profile or extension of OGC CDB 1.x could be created that would support the same GeoPackage containers and level of detail groupings, while conforming to the OGC CDB 1.x conceptual model. Using this approach could bring in some concepts of CDB X into OGC CDB 1.x and make the transition easier to a future version of OGC CDB.

NOTE	We do not recommend supporting more than one tiling scheme in a version of CDB, as this choice is foundational to how data layers are processed and stored and accessed.

13.10. GeoPackage Tile Matrix Set extension

The experiments completed in this effort made use of a proposed GeoPackage Tile Matrix Set extension defined and initially tested in the 2019 OGC Vector Tiles Pilot Phase 2 [https://www.ogc.org/projects/initiatives/vtp2], which enables support for the GNOSIS Global Grid [https://maps.ecere.com/ogcapi/tileMatrixSets/GNOSISGlobalGrid]. This extension has not yet been adopted as an official OGC GeoPackage extension, but is on the GeoPackage Standard Working Group (SWG) roadmap. The current draft is available from here: GeoPackage Two Dimensional Tile matrix Set (TMS) extension (Draft) [https://gitlab.com/imagemattersllc/ogc-vtp2/-/tree/master/extensions/14-tile-matrix-set.adoc].

NOTE	Even though most current software do not yet support the GeoPackage TMS extension, the level 0 data (e.g. for the BlueMarble imagery sample GeoPackages) will still work e.g. in QGIS, as the GNOSIS Global Grid does not use variable widths at level 0.

13.11. Tiled Coverage Data

The GeoPackage Tiled Gridded Coverage extension [http://docs.opengeospatial.org/is/17-066r1/17-066r1.html] would be ideal for storing tiled coverage data. However that Standard has drawbacks that need to be addressed.

Current limitations include:

- Only single channel data is allowed in the coverage extension. Many CDB coverages use more than one channel (Imagery, Raster Materials, etc.). The alternative would be to store CDB X coverage data using different GeoPackage concepts, such as tiles for imagery, coverages for elevation, and related tables for raster material data.
  - Current change request: http://ogc.standardstracker.org/show_request.cgi?id=662
- GeoTiff data only supports using 32-bit floating point data. In OGC CDB 1.x, GeoTiff files are used to store 8 bit unsigned and 8, 16, or 32 bit signed binary data as well. Therefore the proposed

OGC CDB 1.2 will also adopt the use of binary 1-bit data elements as well.

    - Current change request: http://ogc.standardstracker.org/show_request.cgi?id=661

- An issue [https://github.com/opengeospatial/geopackage/issues/551] was filed asking for better clarity about whether 16-bit PNG encoding is signed or unsigned

Multiple sample GeoPackages using TMS / GNOSISGlobalGrid are described and linked below in the results of the experiments.

The following are recommendations and suggested additional discussion topics. These recommendations and discussion topics resulted from the Tiling sub-groups discussion on an enhanced tiling model for CDB X and the potential impacts on the various data types (layers) in the current CDB standard and existing CDB data stores.

13.11.1. Elevation min/max

CDB X needs to continue supporting the Min/Max Elevation component concept. In order to reduce the number of files and complexity, the `recommendation` is to move the minimum and maximum elevation values for the gridded elevation coverage contained in a tile to the tile metadata.

NOTE	The MinElevation and MaxElevation components are part of the MinMaxElevation dataset whose purpose is to provide a CDB conformant data store with the necessary data and structure to achieve a high level of determinism in computing line-of-sight intersections with the terrain. The values of each component are with respect to WGS-84 reference ellipsoid.

13.11.2. Image Compression - JPEG

`Recommendation`: That loss-less and lossy image compression solutions be explored for use in CDB X. Any such solutions are not viewed as a replacement for JPEG 2000 but instead as alternatives. This could be accomplished by submitting a change request for the OGC GeoPackage standard that provides guidance and requirements for support of other image formats beyond PNG and JPG. The sub-group identified a potential candidate: FLIF - Free Lossless Image Format [https://flif.info/], although this format looks to be relatively slow as well.

NOTE	JPEG-2000 has very high compression, even in lossless mode, and there are multiple open-source implementations. However, performance can be extremely slow and non-optimal for all use cases.

13.11.3. Materials

`Recommendation`: CDB X needs to support material data to provide the same functionality as CDB 1.x. To also reduce the number of files, this can be accomplished by putting all the raster material data (including material table) in a single CDB data layer in GeoPackage, perhaps using the related tables extension. The subgroup did have some discussion on what "materials" means in the CDB 1.x context. Materials in current CDB have to do with the physical substance of a feature that can then be used to simulate the emmisive or reflective properties of a feature in wavelengths of the electromagnetic spectrum other than what the human eye senses. These are for non-visualization

use cases or special visualization such as IR or Radar. The subgroup did also discuss for the possible need for CDB X to provide guidance on using Physically-Based Rendering (PBR) to support the visualization/rendering use case. glTF, I3S, and 3D Tiles all support PBR.

13.12. Tiled Vector Data

To tile vector data (including points referencing 3D models), draft GeoPackage extensions defined during the *OGC Vector Tiles pilots* were used:

1. GeoPackage Vector Tiles [https://gitlab.com/imagemattersllc/ogc-vtp2/-/blob/master/extensions/1-vte.adoc] The draft GeoPackage Tiled Vector Data extension defines the rules and requirements for encoding tiled feature data (aka "vector tiles") into a GeoPackage data store.
2. GeoPackage Vector Tiles Attributes Extension [https://gitlab.com/imagemattersllc/ogc-vtp2/-/blob/master/extensions/4-vtae.adoc] This draft extension defines a relationship between features contained in a tiled layer and tiles containing those features.
3. MapBox Vector Tiles extension [https://gitlab.com/imagemattersllc/ogc-vtp2/-/blob/master/extensions/2-mvte.adoc] The draft GeoPackage Mapbox Vector Tiles extension defines the rules and requirements for encoding vector tiles in a GeoPackage data store as Mapbox Vector Tiles.

As an alternative to encoding tiled vector data as Mapbox Vector Tiles, some experiments used the GNOSIS Map Tiles [http://docs.opengeospatial.org/per/18-025.html#GMTSpecs] encoding, specified by Ecere in OGC Testbeds 13 and 14.

`Recommendation`: Although the use of non-tiled vector data layers (e.g. storing the geometry as WKB in GeoPackage features tables) should also be specified in the CDB Standard, the use of tiled vector data extension should be allowed. In particular, tiling vector data is essential for dealing with features spanning a very large geospatial extent, such as coastlines (e.g. OpenStreetMap ways tagged with `natural=coastline` [https://wiki.openstreetmap.org/wiki/Coastline]).

NOTE	Tiling of large single features was not tested in experiments done by the Vector data group. That group focused on testing the performance of a large number of features stored using indexes but not tiled inside GeoPackages.

One observation is that even if the vector geometry is tiled and organized into multiple GeoPackages, it might be useful to support storing the data attributes separately only at the top-level (level 0) tiles, or in a single GeoPackage storing only data attributes, to avoid duplication of that information at each grouping level.

13.13. Tiled 3D Models

Two approaches to include 3D Models in GeoPackages were tested. These tests included the textures used by those models.

- Approach A) One in which 3D models are individually stored in a single table and referenced and placed by information tiled vector data (points).
- Approach B) The other in which batched 3D models are the content of the tiles.

Results of experiments for both approaches are found below in the experiments description.

13.13.1. 3D Models Extension

As part of defining a draft GeoPackage extension for 3D models, rows were added to the `gpkgext_extensions` table to identify all tables set up for this proposed extension.

For all of these tables, the `extension_name` was configured to be `ecere_3d_models`. The definition was http://github.com/ecere/gpkgext-3dmodels, and the `scope` was `read-write`.

The tables registered with this extension were:

- `gpkgext_3d_models` (only used for the reference points Approach A).
- The individual tiles tables for batched 3D models (only used for Approach B).
- `gpkgext_textures` for shared textures.

13.13.2. A) Referenced 3D models with placement information

For Approach A, best suited for geo-typical models, a single table per GeoPackage (`gpkgext_3d_models`) was used to define one model per row. These were blobs within the `model` field. A `format` field was used to specify the format. Both glTF and E3D {**Need a short definition for context**} were used in the experiments. The `name` field enabled specifying a name for the model. The `lod` field can optionally be used to distinguish between multiple level of details for the model, or left NULL if only a single version exists. The combination of `name`, `lod` and `format` must be unique.

The `model::id` field of the attributes table for the 3D models referencing points (which would also contain the point geometry in a non-tiled approach) references the `id` (primary key) of the `gpkgext_3d_models` table. The attributes table may also contain additional fields for scaling and orienting the models:

- `model::scale`; or `model::scaleX`, `model::scaleY` and `model::scaleZ` for non-uniform scaling
- `model::yaw`, `model::pitch` and `model::roll`

These attributes duplicate the CDB fields `AO1`, `SCALx`, `SCALy`, `SCALz` (and in a sense `MODL` as well). However these values are intended to be defined in a non-CDB specific manner within a generic 3D Models extension for GeoPackage.

The following SQL is used to create the `gpkgext_3d_models` table:

```
CREATE TABLE gpkgext_3d_models (
   id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
   name TEXT NOT NULL,
   lod INTEGER,
   format TEXT NOT NULL,
   model BLOB,
   CONSTRAINT unique_models UNIQUE(name, lod, format));
```

Example `gpkgext_3d_models`:

```
id  name                lod  format  model
--  ------------------  ---  ------  -----
1   coniferous_tree01       glb     glTF
2   palm_tree01             glb     glTF
```

Sample SQL table creation for attributes table referencing the 3D models:

```
CREATE TABLE attributes_Trees (
   id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
   AO1 REAL, CNAM TEXT, RTAI INTEGER,
   SCALx REAL, SCALy REAL, SCALz REAL,
   AHGT TEXT, BBH REAL, BBL REAL, BBW REAL, BSR REAL,
   CMIX INTEGER, FACC TEXT, FSC INTEGER, HGT REAL,
   MODL TEXT,
   `model::id` INTEGER,
   `model::yaw` REAL,
   `model::scale` REAL)
```

13.13.3. B) Batched 3D Models tiles

For Approach B, best suited for geo-specific models, a single model covers a whole tile, all 3D models from the data layer found within that tile are batched, and is stored in a tiles table much like raster or vector tiles (as a glTF blob in the `tile_data` field). This is closer to the 3D Tiles / One World Terrain approach, and could potentially also combine both 3D Terrain and 3D Models (though ideally keeping them as distinct nodes within the model). Such an approach may facilitate transition between CDB-X and OWT.

Because GeoPackage does not define a generic mechanism for specifying the encoding of `tile_data` (this has previously been suggested that this would be a good field to add to the `gpkg_contents` table), the encoding of the 3D model must be deducted from the content of the blob. Fortunately, both glTF and E3D feature a header signature that facilitates this. The `3d-models` type was introduced for specifying the `data_type` column of the `gpkg_contents` table.

The translation origin of the model, as well as its orientation, is implied from the center of the tile (from the tile matrix / tile matrix set) for which it is defined. The model is defined in the 3D cartesian space where (0, 0, 0) lies at that center, sitting directly on the WGS84 ellipsoid {*This is strange wording. In 3D geo, isn't tis typically called "clamping" or "clamped"?}, and oriented so that by default the model appears upright with its X axis pointing due East, its Z axis pointing North, and its Y axis pointing away from the center of the Earth. In other words, this is equivalent to having a single point situated at the center of the tile in the referenced 3D points approach.

The height of the individual features, such as buildings, within the batched models tile models was adjusted to match the elevation model. However, each separate feature from CDB is encoded in the model as a separate node to facilitate re-adjusting it to new elevation.

13.13.4. Textures table

The textures table has the following fields:

- `id`: integer primary key
- `name`: The filename used in the model to refer to the texture
- `width`: width of the texture
- `height`: height of the texture
- `format`: e.g. "png"
- `texture`: blob containing the texture data

The combination of `name`, `width`, `height` and `format` must be unique.

The following SQL statement is used to create the table:

```
CREATE TABLE gpkgext_textures (
   id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
   name TEXT NOT NULL,
   width INTEGER NOT NULL,
   height INTEGER NOT NULL,
   format TEXT NOT NULL,
   texture BLOB,
   CONSTRAINT unique_textures UNIQUE(name, width, height, format));
```

```
id  name   width  height  format  texture
--  -----  -----  ------  ------  -------
1   1.png  512    512     png       PNG
2   1.png  256    256     png       PNG
3   2.png  512    512     png       PNG
4   2.png  256    256     png       PNG
```

13.14. Ecere GeoPackage Tiling Experiments

Ecere conducted multiple experiments during the sprint using tiling and packaging data sourced from CDB 1.x content.

- Experiment 1: The objective of the first experiment was to share an example of how to store CDB data in a GeoPackage following the GNOSIS Global Grid. This Experiment used the draft GeoPackage TileMatrixSet and vector tiles extensions, as well as describing this content using the proposed `cdb.json` schema. A sample CDB datastore from Presagis for Camp Pendleton California was used for this experiment.
- Experiment 2: The objective of the second experiment focused on demonstrating how to distribute data covering a large area in multiple GeoPackages using a pre-determined LOD grouping setting. The larger San Diego CDB dataset provided by CAE was used for this Experiment as well as for Experiment 3.

- Experiment 3: The objective of the third experiment was investigating storing 3D models inside the GeoPackages using the two different approaches described above. Approach A is very similar to the CDB 1.x data model where all 3D models are stored in a single table and referenced from points stored in tiled vector data. In Approach B, 3D models are batched on a per tile basis and stored in the `tile_data` blob of GeoPackage tiles table. The local origin of the 3D model corresponds to the center of the tile in which it is stored, assuming a coordinate system whose axes are aligned with East, North and Up. Approach B is more similar to the one used for 3D Tiles and One World Terrain.

These were originally described as separate experiments on the CDB X concept wiki [https://github.com/sofwerx/cdb2-concept/wiki]. However, below the combined final results of producing a prototype CDB X for the full San Diego CDB are presented. These results cover tiling the imagery, elevation, vector data and 3D models for the full San Diego CDB.

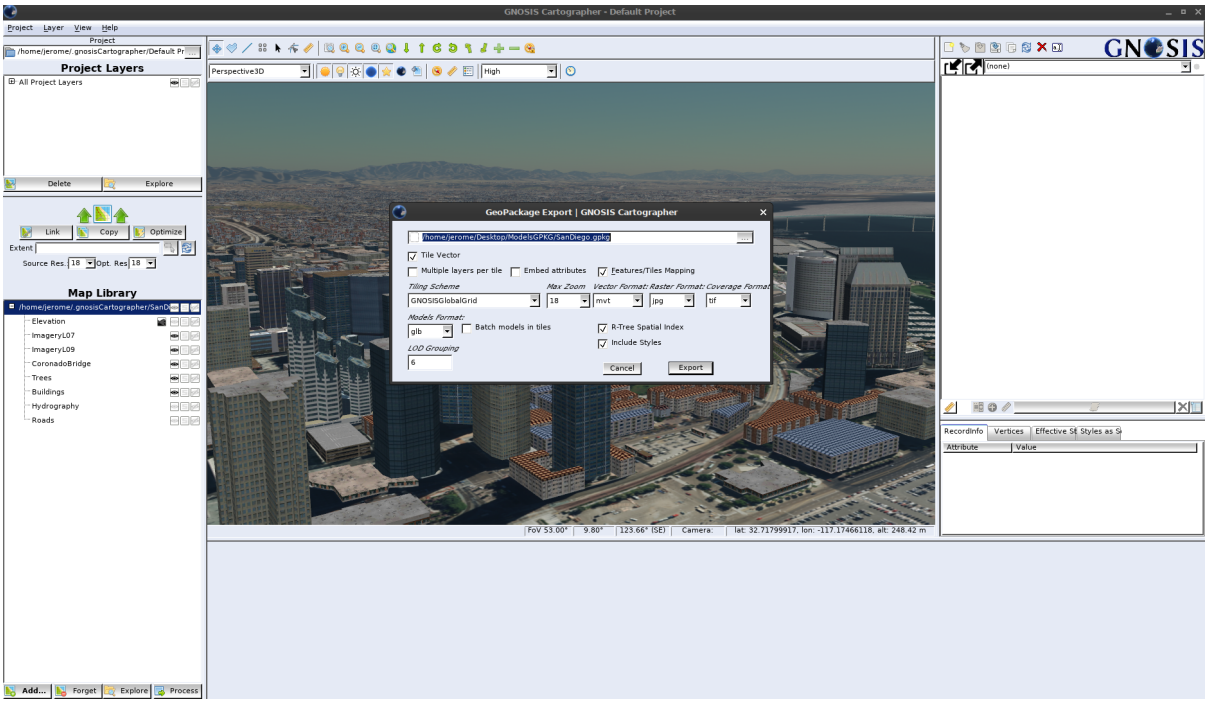


Figure Mtd_Ph3 - 31. Ecere GeoPackage / CDB X Export dialog and options in Ecere's GNOSIS Cartographer

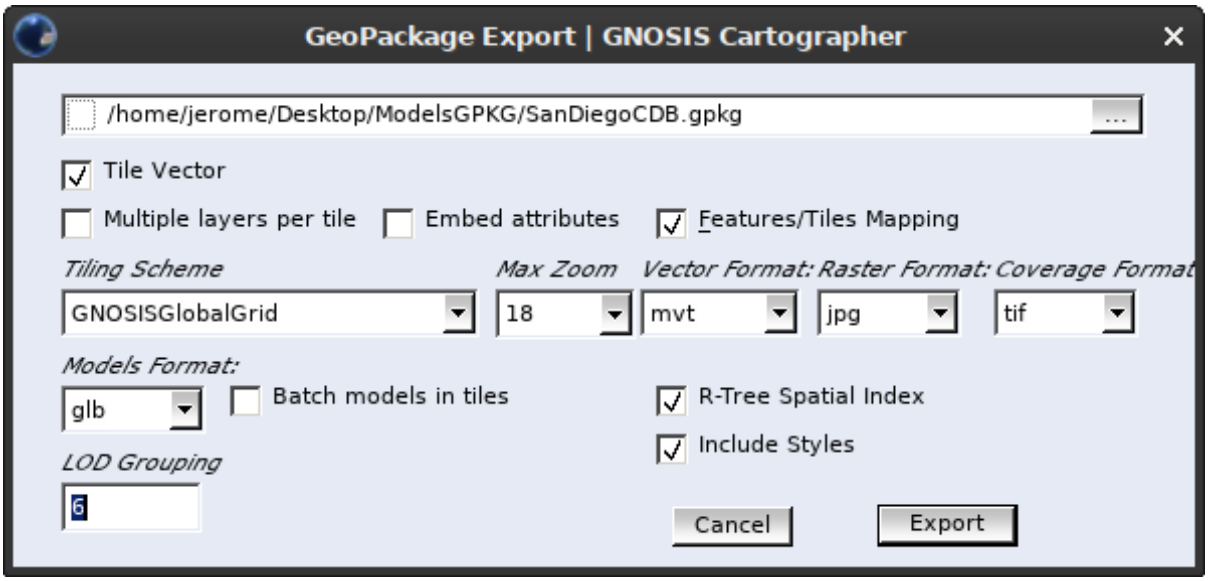


Figure Mtd_Ph3 - 32. Ecere GeoPackage / CDB X Export dialog and options close up

The results produced were sourced from the CAE San Diego CDB 1.x, and unless otherwise specified cover the entire dataset.
An additional data layer was sourced from the NASA Visible Earth Blue Marble [https://visibleearth.nasa.gov/collection/1484/blue-marble].
These are also available from the CDB SWG / CDB X Tech Sprint folder [https://portal.ogc.org/index.php?m=projects&a=view&project_id=466&tab=2&artifact_id=95315].

MVT: Mapbox Vector Tiles (specifications [https://docs.mapbox.com/vector-tiles/reference/])

GMT: GNOSIS Map Tiles (specifications [http://docs.opengeospatial.org/per/18-025.html#GMTSpecs])

E3D: Ecere 3D Models (specifications [http://docs.opengeospatial.org/per/18-025.html#E3DSpecs])

(from *CityGML and Augmented Reality* Engineering Report [http://docs.opengeospatial.org/per/18-025.html])

13.14.1. San Diego CDB layers packaged together

Data / Link	LOD Grouping	Models Encoding	Imagery Encoding	Coverage Encoding	Vector Encoding	Uncompressed size	Compressed Size
(*cdb.json* [https://portal.ogc.org/files/?artifact_id=95378])	***As a single GeoPackage***						
In the textures table for this GeoPackage, the texture names are prefixed by an extra directory to differentiate the numbered textures used in the different 3D models data layers. However the reference from the 3D models was not yet updated to reflect this.							
San Diego [https://data.ogc.org/2020/11/SanDiego.gpkg.7z]	Single File	Binary glTF	JPEG	PNG	MVT	12.8 GiB	9.8 GiB
San Diego [https://data.ogc.org/2020/11/SanDiego.gpkg]	Single File	E3D	JPEG	GMT	GMT	10.3 GiB	
(*cdb.json* [https://portal.ogc.org/files/?artifact_id=95366])	***Grouping tiles of 5 LODs per GeoPackage***						

In this approach, the textures were stored in a separate folder to avoid repeating them in each separate GeoPackage.							
San Diego [https://data.ogc.org/2020/11/SanDiego.7z]	5	Binary glTF	JPEG	PNG	MVT	14.6 GiB	9.9 GiB
San Diego (subset) [https://portal.ogc.org/files/?artifact_id=95370]	5	Binary glTF	JPEG	PNG	MVT	1.4 GiB	217.5 MiB

13.14.2. San Diego CDB packaged as separate layers

Top-level *cdb.json* index [https://portal.ogc.org/files/?artifact_id=95345]

Elevation

Data / Link	LOD Grouping	Coverage Encoding	Uncompressed size	Compressed Size
Elevation [https://portal.ogc.org/files/?artifact_id=95352]	6	PNG / 16-bit integer	483 MiB	472 MiB
Elevation [https://portal.ogc.org/files/?artifact_id=95328]	6	GMT / 16-bit integer	370.8 MiB	365.2 MiB
Elevation [https://data.ogc.org/2020/11/ElevationTIF.7z]	6	GeoTIFF / 32-bit float	1.4 GiB	1.3 GiB

Imagery

NOTE	The imagery in these GeoPackages is lossy. `Recommendation`: Allow the use of JPEG-2000 and/or additional lossless formats more compact than PNG in GeoPackages.

Data / Link	LOD Grouping	Imagery Encoding	Uncompressed size	Compressed Size

Blue Marble [https://data.ogc.org/2020/11/BlueMarble.7z]	7	JPEG	1.6 GiB	1.4 GiB
Imagery (Medium) [https://data.ogc.org/2020/11/ImageryMedium.7z]	7	JPEG	4.9 GiB	4.7 GiB
Imagery (High) [https://data.ogc.org/2020/11/ImageryHigh.7z]	7	JPEG	4.5 GiB	4.3 GiB

Vector data

Data / Link	LOD Grouping	Vector Encoding	Uncompressed size	Compressed Size
Hydrography [https://portal.ogc.org/files/?artifact_id=95348]	Single File	Mapbox Vector Tiles	424 KiB	
Roads [https://portal.ogc.org/files/?artifact_id=95350]	Single File	Mapbox Vector Tiles	69.1 MiB	17.6 MiB
Airport Lights [https://portal.ogc.org/files/?artifact_id=95346]	Single File	Mapbox Vector Tiles	144 KiB	
Hydrography [https://portal.ogc.org/files/?artifact_id=95330]	Single File	GNOSIS Map Tiles	248 KiB	
Roads [https://portal.ogc.org/files/?artifact_id=95331]	Single File	GNOSIS Map Tiles	38 MiB	23.4 MiB
Airport Lights [https://portal.ogc.org/files/?artifact_id=95329]	Single File	GNOSIS Map Tiles	148 KiB	

3D Models

Using referencing placement points

Data / Link	LOD Grouping	Models Encoding	Vector Encoding	Uncompressed size	Compressed Size
Buildings [https://portal.ogc.org/files/?artifact_id=95351]	Single File	Binary glTF	Mapbox Vector Tiles	2.9 GiB	321.3 MiB
Trees [https://portal.ogc.org/files/?artifact_id=95349]	Single File	Binary glTF	Mapbox Vector Tiles	2 MiB	
Coronado Bridge [https://portal.ogc.org/files/?artifact_id=95347]	Single File	Binary glTF	Mapbox Vector Tiles	272 KiB	
Buildings [https://portal.ogc.org/files/?artifact_id=95340]	Single File	E3D	GNOSIS Map Tiles	560.8 MiB	332.8 MiB
Trees [https://portal.ogc.org/files/?artifact_id=95339]	Single File	E3D	GNOSIS Map Tiles	1.8 MiB	
Coronado Bridge [https://portal.ogc.org/files/?artifact_id=95338]	Single File	E3D	GNOSIS Map Tiles	196 KiB	
Buildings [https://portal.ogc.org/files/?artifact_id=95344]	4	Binary glTF	Mapbox Vector Tiles	4.1 GiB	446.6 MiB
Buildings [https://portal.ogc.org/files/?artifact_id=95343]	5	Binary glTF	Mapbox Vector Tiles	3.7 GiB	409.2 MiB
Buildings [https://portal.ogc.org/files/?artifact_id=95342]	7	Binary glTF	Mapbox Vector Tiles	2.9 GiB	321.3 MiB

Trees [https://portal.ogc.org/files/?artifact_id=95341]	6	Binary glTF	Mapbox Vector Tiles	2.9 MiB	1.2 MiB

NOTE The larger size for grouping by 4 and 5 LODs is mainly a result of lower resolution models being duplicated in higher level grouping GeoPackages.

NOTE Except for the directory / file naming, the 7 LODs buildings are equivalent to 6 LODs since there are only 6 LODs of buildings in the dataset. Similarly, the 6 LODs trees are equivalent to any other LOD groupings, since there is only 1 LOD of trees in the dataset.

Batching models per tile

Data / Link	LOD Grouping	Models Encoding	Vector Encoding	Uncompressed size	Compressed Size
Buildings [https://portal.ogc.org/files/?artifact_id=95337]	Single File	Binary glTF	Mapbox Vector Tiles	3.9 GiB	512.4 MiB
Trees [https://portal.ogc.org/files/?artifact_id=95335]	Single File	Binary glTF	Mapbox Vector Tiles	91 MiB	2.3 MiB
Coronado Bridge [https://portal.ogc.org/files/?artifact_id=95336]	Single File	Binary glTF	Mapbox Vector Tiles	576 KiB	
Buildings [https://portal.ogc.org/files/?artifact_id=95334]	Single File	E3D	GNOSIS Map Tiles	417.3 MiB	339.8 MiB
Trees [https://portal.ogc.org/files/?artifact_id=95333]	Single File	E3D	GNOSIS Map Tiles	3.3 MiB	
Coronado Bridge [https://portal.ogc.org/files/?artifact_id=95332]	Single File	E3D	GNOSIS Map Tiles	256 KiB	

NOTE	The uncompressed GeoPackages are more compact because the E3D models feature internal LZMA compression.

NOTE	The model is encoded in the `tile_data` field of the tiles table. This approach is best suited for geo-specific models, and can be used directly as the glTF payload of a `.b3dm` 3D Tile (with proper transformation matrix in JSON tileset).

13.14.3. OGC API access demo

The SanDiegoCDB data store can be found at this address: https://maps.ecere.com/ogcapi/collections/SanDiegoCDB

The San Diego CDB data from the GNOSIS data store can be accessed directly through the GNOSIS Map Server. This includes rendering maps, downloading coverages, accessing as tiles in different tiling schemes, accessing individual vector features, retrieving them as (re-merged) GeoJSON, visualizing them on GeoJSON.io, accessing as 3D Tiles tilesets or individual tiles, also supporting retrieving batched 3D tiles as binary glTF and so on. These delivery capabilities demonstrate that tiled data actually supports a wide range of use cases. More details about accessing this data via the OGC API {**Which OGC API? I am assuming features.**} can be found in the OGC Interactive Simulation and Gaming Sprint Engineering Report{**Do we need a link?**} .

The prototype CDB X split GeoPackages can be accessed directly at https://maps.ecere.com/ogcapi/collections/SanDiegoLayers [address]:

However at this point, the map server and visualization client do not present this as a unified data source. As such, the tiles structure and individual GeoPackages are individually accessible instead.

13.14.4. Visualization

Ecere's GNOSIS 3D visualization tools can currently visualize the individual CDB X/GeoPackage elevation and imagery directly. As for the server, however, the split GeoPackages are not yet unified as a single data source. Accessing and visualizing the 3D models from the GeoPackage tables remains to be implemented.

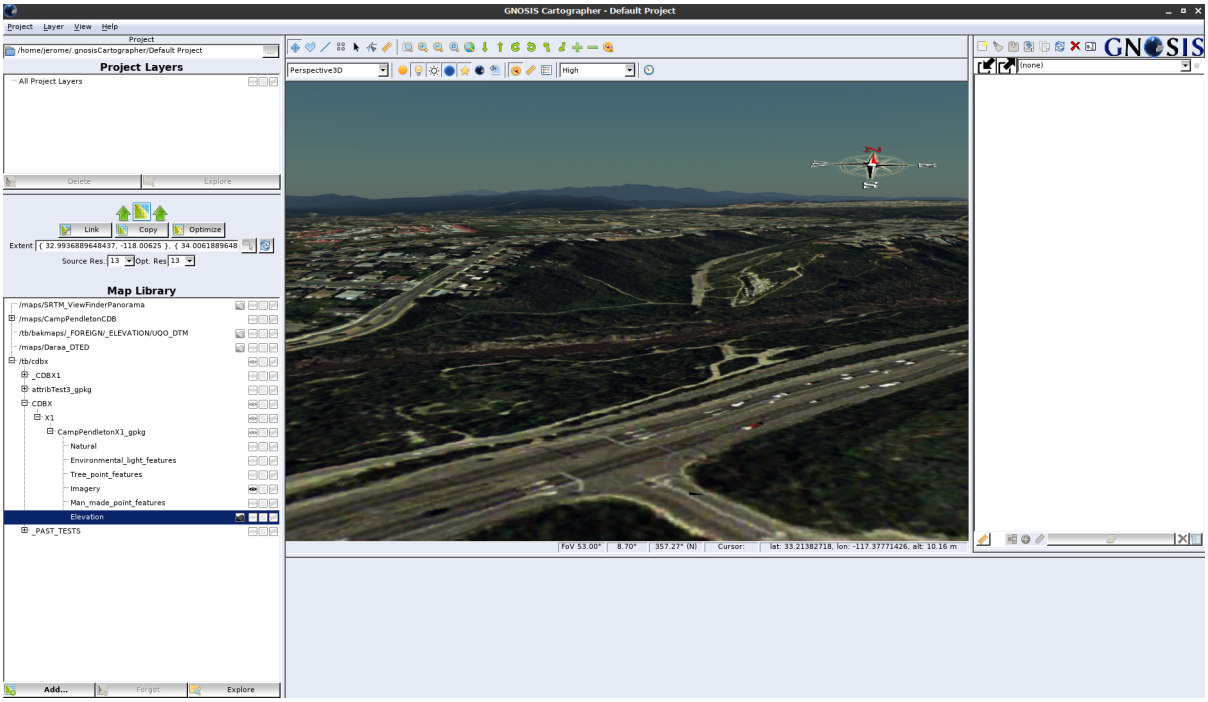


Figure Mtd_Ph3 - 33. Visualizing Camp Pendleton CDB X/GeoPackage in Ecere's GNOSIS Cartographer.

The following screenshots are visualization of the intermediate GNOSIS data store used to generate the CDB X using the same tiling scheme. Part of this effort was accomplished during the OGC Interoperable Simulation and Gaming Sprint [https://www.ogc.org/projects/initiatives/isg-sprint]. A video [https://www.youtube.com/watch?v=gyaQjqy0N8g] was also published.

In addition to the San Diego CDB dataset, worldwide elevation data from Viewfinder Panoramas [http://viewfinderpanoramas.org/] by Jonathan de Ferranti and imagery from NASA Visible Earth's Blue Marble [https://earthobservatory.nasa.gov/features/BlueMarble] are used outside of the extent covered by the San Diego dataset.

Figure Mtd_Ph3 - 34. San Diego CDB data visualized in Ecere's GNOSIS Cartographer (cape)

Figure Mtd_Ph3 - 35. San Diego CDB data visualized in Ecere's GNOSIS Cartographer (hotels and palm trees)

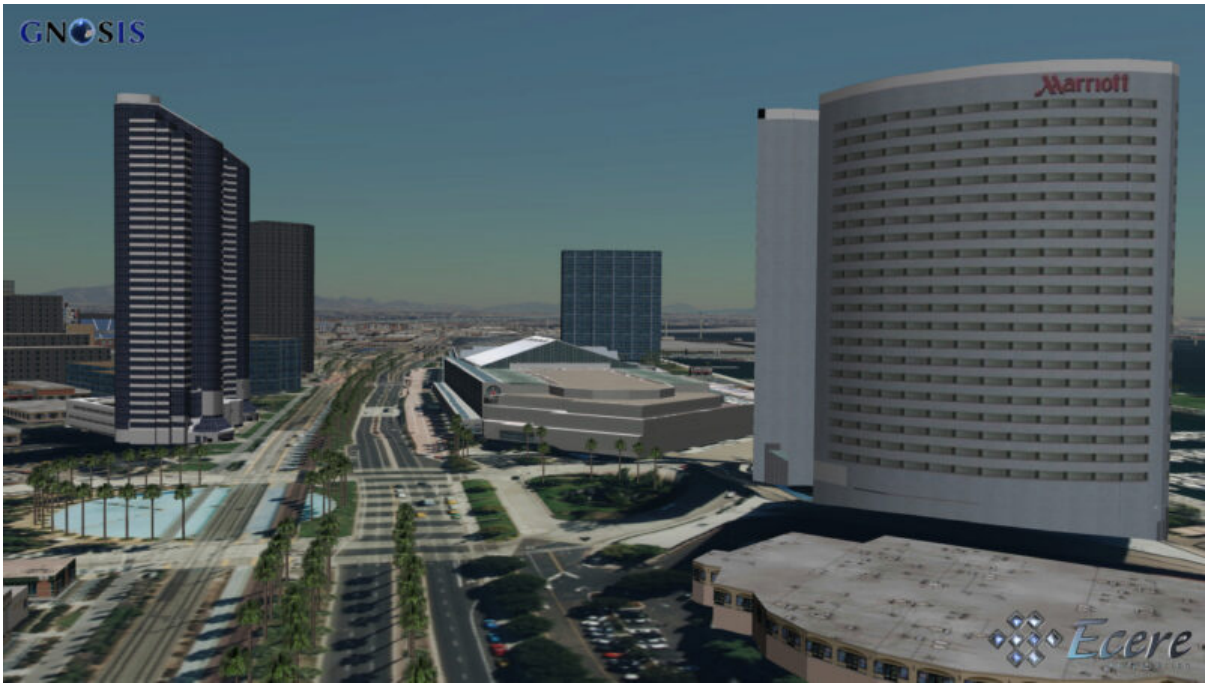


Figure Mtd_Ph3 - 36. San Diego CDB data visualized in Ecere's GNOSIS Cartographer (skyscrapers)

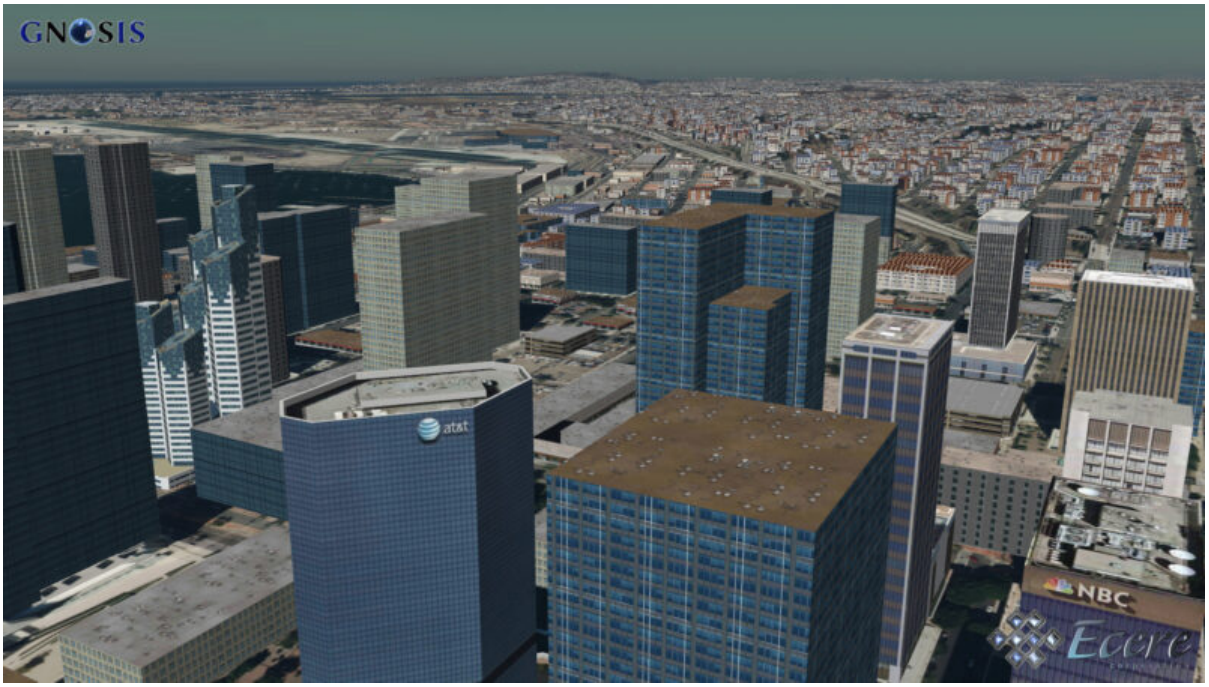


Figure Mtd_Ph3 - 37. San Diego CDB data visualized in Ecere's GNOSIS Cartographer (Coronado bridge)

Figure Mtd_Ph3 - 38. San Diego CDB data visualized in Ecere's GNOSIS Cartographer (airstrip)

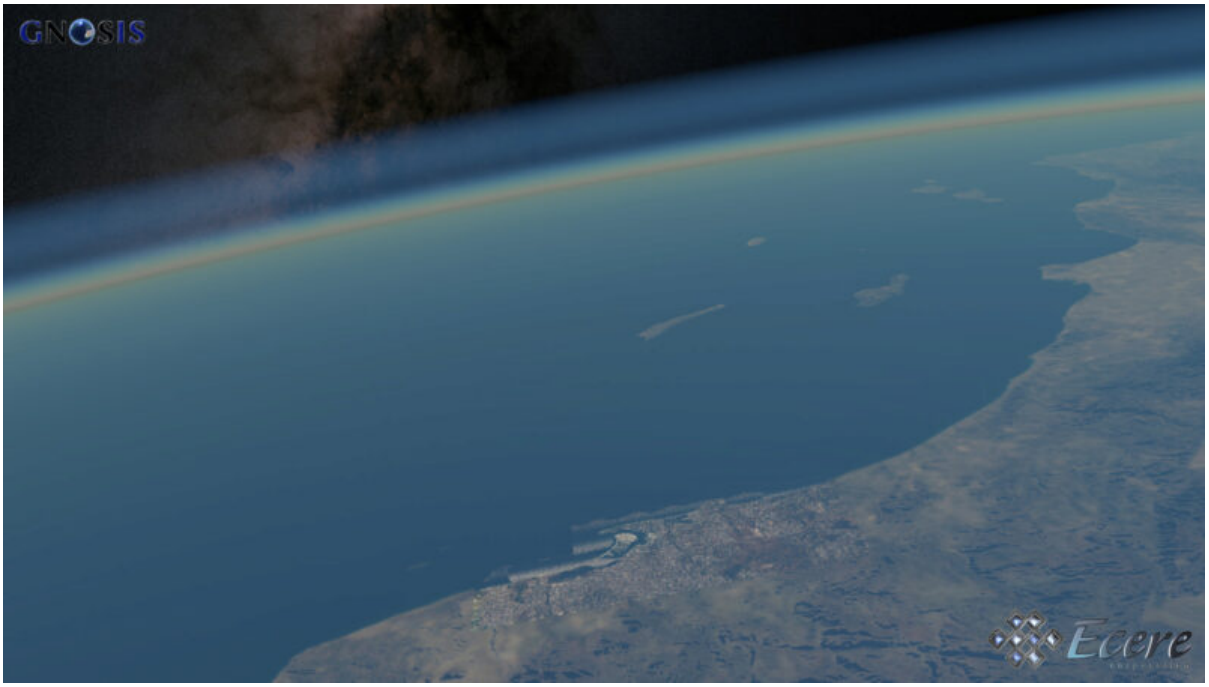


Figure Mtd_Ph3 - 39. San Diego CDB data visualized in Ecere's GNOSIS Cartographer (high above, showing 3D globe)

This last image features ESA Gaia's Sky in colour (Gaia Data Processing and Analysis Consortium (DPAC); A. Moitinho / A. F. Silva / M. Barros / C. Barata, University of Lisbon, Portugal; H. Savietto, Fork Research, Portugal.) CC BY SA 3.0.

13.14.5. Cesium JS / 3D Tiles demo

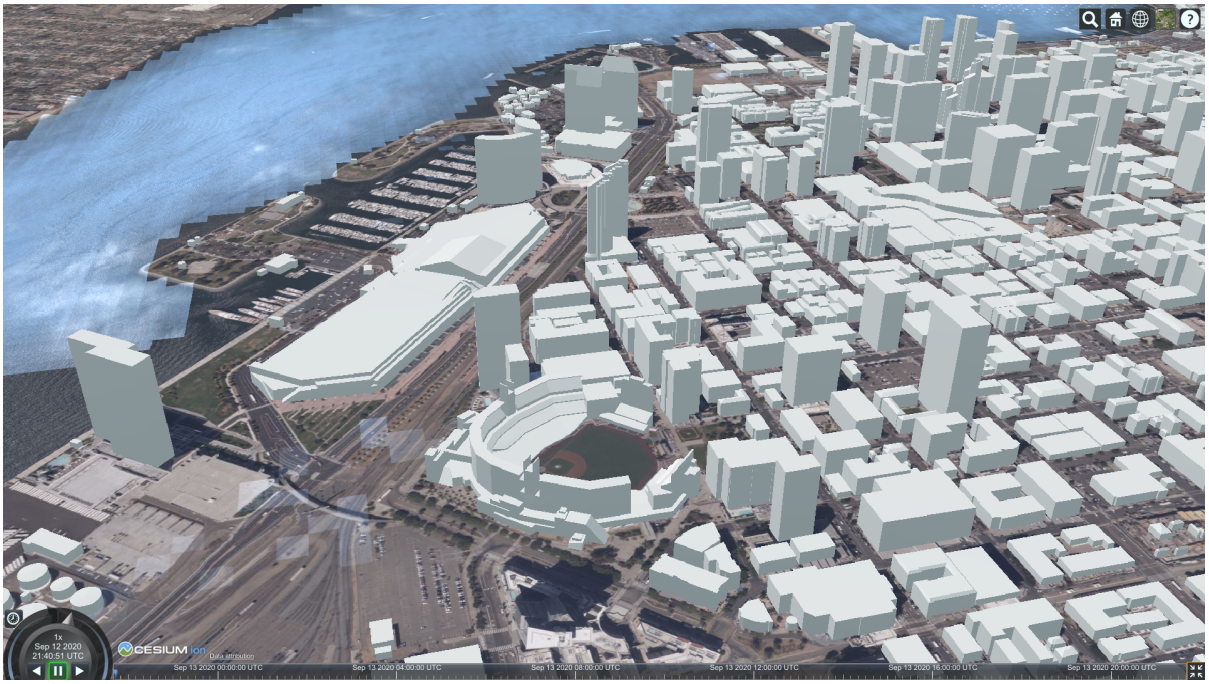


Figure Mtd_Ph3 - 40. San Diego CDB (before implementing support for textures in generated 3D Tiles)

Figure Mtd_Ph3 - 41. CesiumJS Client accessing San Diego CDB data as 3D Tiles from Ecere's GNOSIS Map Server (Petco Park)

The following JavaScript code, which can simply be copied to the Cesium Sand Castle [https://sandcastle.cesium.com/], can be used to visualize the data as 3D Tiles, including textures. The code sets up the buildings, trees as well as the Coronado Bridge, together with the Cesium world terrain. One limitation is that the generated tileset is still missing multiple level of details, therefore visualizing a large area will be quite slow.

```
var worldTerrain = Cesium.createWorldTerrain({ requestWaterMask: true,
requestVertexNormals: true });
var viewer = new Cesium.Viewer("cesiumContainer", { terrainProvider: worldTerrain });
var scene = viewer.scene;
var trees = scene.primitives.add(new Cesium.Cesium3DTileset(
   { url:
"https://maps.ecere.com/ogcapi/collections/SanDiegoCDB:Trees/3DTiles/tileset.json"
}));
var bridge = scene.primitives.add(new Cesium.Cesium3DTileset(
   { url:
"https://maps.ecere.com/ogcapi/collections/SanDiegoCDB:CoronadoBridge/3DTiles/tileset.
json" }));
var buildings = scene.primitives.add(new Cesium.Cesium3DTileset(
   { url:
"https://maps.ecere.com/ogcapi/collections/SanDiegoCDB:Buildings/3DTiles/tileset.json"
}));
```

This 3D Tiles distribution is currently being generated from the GNOSIS Data Store / E3D models. Support to stream as 3D Tiles straight from CDBX GeoPackages should also be achievable.

13.14.6. Future work

- Support for visualizing 3D models directly from the CDB X/GeoPackages dataset in GNOSIS

Cartographer client.

- Support for GNOSIS Map Server streaming 3D models directly from CDB X/GeoPackage.
- Support for unifying split GeoPackages making up the CDB X dataset as a single data source.
- Attribution per model within the single tile model. This is supported directly in E3D for triangular face-level attribution (it was clarified that glTF2 does not supports this, and extensions were considering vertex rather than face attributions).

13.15. FlightSafety GeoPackage Tiling Experiments

Setup: The data used for these experiments are primarily freely available, and include the following:

- Blue Marble (NASA) that was georeferenced using GDAL - https://visibleearth.nasa.gov/collection/1484/blue-marble
- The high resolution inset is from USGS downloads of Central Park in New York City

Tiling Scheme: The tiling scheme uses the GNOSIS Global Grid [https://maps.ecere.com/ogcapi/tileMatrixSets/GNOSISGlobalGrid] (using TMS extension — https://gitlab.com/imagemattersllc/ogc-vtp2/-/blob/master/extensions/14-tile-matrix-set.adoc). FlightSafety used the same type of json file {**Perhaps a bit moreinfo and a link back?**} that Ecere used in their experiment.

LOD Grouping The grouping was pre-set per experiment. The groups were calculated from the highest LOD, back to coarser LODs. For example, if there are 7 LODs (0-6) and a grouping of 4, then LODs 3 through 7 are in one GeoPackage, and LODs 0 through 2 are in another GeoPackage.

Directory and Naming Scheme Each top level tile is within a directory that encodes the LOD, the row (rows are counted from the top, so north to south), and the column (longitude west to east). For example, "L0_R1_C2". Each tile directory contains one GeoPackage file (for example "Imagery_L0_L2_R1_C2.gpkg") and all the tile directories that refine this area (such as "L3_R9_C22"). There were two intentions to this directory structure:

- Limit the number of files in a directory (to keep from running into OS limitations).
- Make it a bit easier to export a portion of the world by hand from one CDB X to another.

13.15.1. FlightSafety Experiment 1

Purpose of Experiment

This experiment was designed to:

- Show how the top levels of the tiling scheme work,
- Show the LOD groupings within multiple GeoPackage files, and
- Show the proposed directory and file naming.

There were eight top level tiles (2 rows and 4 columns) and all GeoPackages that refine one of these tiles are under that tile's directory structure.

Processing

This experiment used the NASA Blue Marble imagery to approximate world-wide imagery at a high level. This provides seven levels of detail of data (L0 to L6). Normally, the GeoPackage files should be larger for efficient use. However to demonstrate the LOD groupings, only four LODs were grouped together. So that tools can view the imagery more easily, the imagery is stored as JPEGs. Originally the thought was to create Jpeg2000 files but checking the results in a tool such as "DB Browser for SQLite" was harder. The content volume for the data used this experiment was around 300 MB.

Data Location

Compressed 7-zip file with test data can be found at: https://portal.ogc.org/files/?artifact_id=95358

13.15.2. FlightSafety Experiment 2

Purpose of Experiment

This experiment was designed to further test the limits of the LOD grouping and directory organization. This experiment is similar to the World CDB X Experiment 1 but with a small higher resolution inset of imagery. Images added were 15m data at LOD level 12 covering New York City and 2 ft imagery covering Central Park on Manhattan Island at LOD level 16.

Processing

The same processing was used as in Experiment 1 but with an LOD grouping of 6. During the sub-groups planning for this experiment, the hyposthesis was that was an ideal balancing size and number of sub-directories ($(2^{6})2$ = 4096 maximum directories within one folder. The maximum LOD for this experiment was 16 (60cm). To find the highest resolution data, look at file CDBX_highres\Imagery\L0_R0_C1\L5_R17_C37\L11_R1120_C2412\Imagery_L11_L16_R1120_C2412.gpkg. The data size for this experiment was almost 1.5 GB.

Data Location

The full compressed file had to be split into two pieces as a multi-part zip file, to enable saving on the OGC portal.

- https://portal.ogc.org/files/?artifact_id=95361
- https://portal.ogc.org/files/?artifact_id=95371

13.15.3. Observations from Experiments 1 and 2

- The file names and directory names are pretty hard to read and understand by looking at the files. However, since the tiles are rarely on a "geocell" boundary, a good naming scheme may not exist.
- Creating the LOD groupings based on the highest LOD of data makes it difficult to add data of a higher resolution later on. This might also make it harder to create "Versions" of the data that have been updated.

- There were a considerable number of directories created with this tiling and naming scheme. In general, there is a 1-to-1 ratio of files to directories.

13.15.4. FlightSafety Experiments 3 and 4

Purpose of Experiment

These experiments utilized two different tiled layers: Imagery and Elevation. The constraints for this experiment were :

- There are two different tiled layers: Imagery and Elevation.
- The data coverage was world-wide, containing 1000m resolution imagery and elevation.
- The directory structure was reworked to reduce the number of directories produced so that it was no longer a 1-to-1 file to directory ratio. To copy over a section of the world, one would need to copy both the GeoPackage and the directory with similar names
- The GeoPackage files were renamed to be **lod_row_col_endlod**.gpkg, to keep the lod/row/column triplet together. For example, Imagery_L4_R9_C6_L6.gpkg

Updated Directory Structure

The directory structure was changed from having each GeoPackage within a directory of the same name (yielding a 1:1 ratio of files to directories) to having a finer resolution GeoPackage in a directory with the coarser tile name. If there is even finer/higher resolution data beyond this GeoPackage, that data will be found in a directory at the same level as the GeoPackage with the tile name that matches most of the GeoPackage filename (except for the end lod value). Pictures of the structure below:

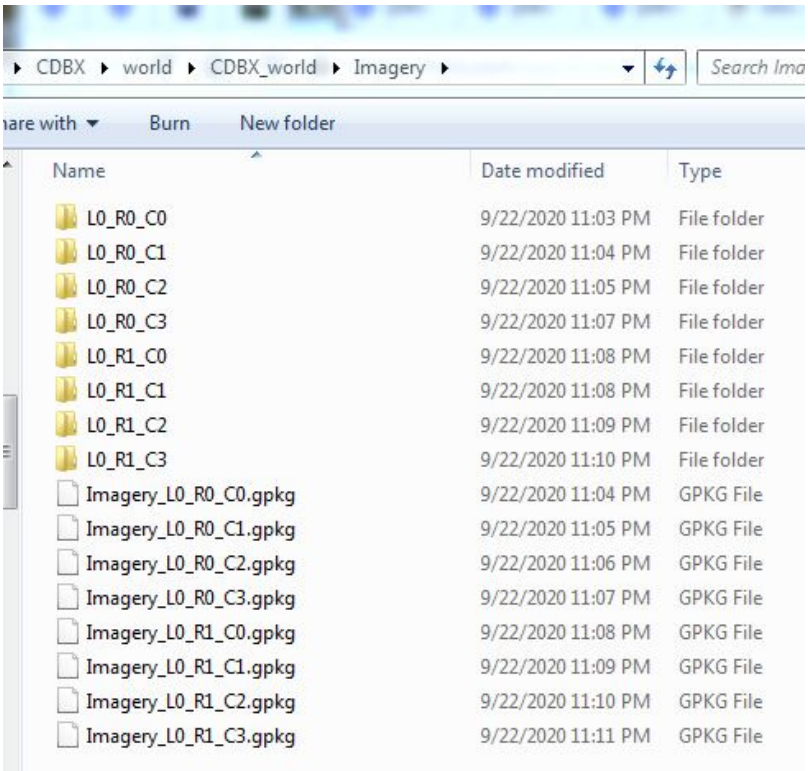


Figure Mtd_Ph3 - 42. Top GeoPackage Level.

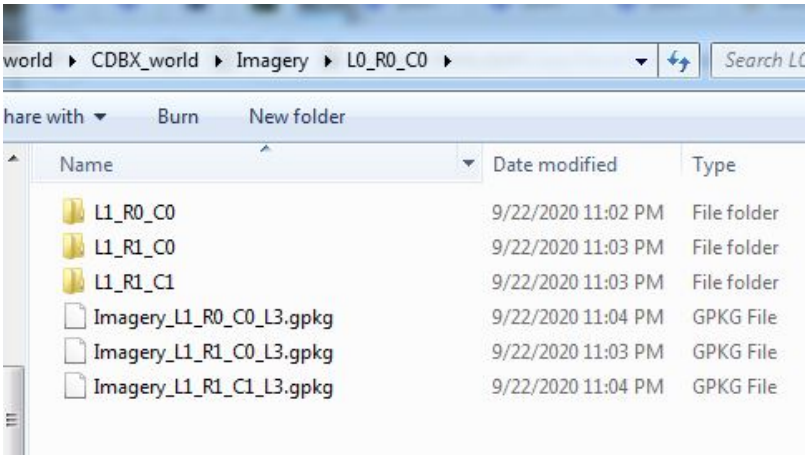


Figure Mtd_Ph3 - 43. Mid-level directory structure.

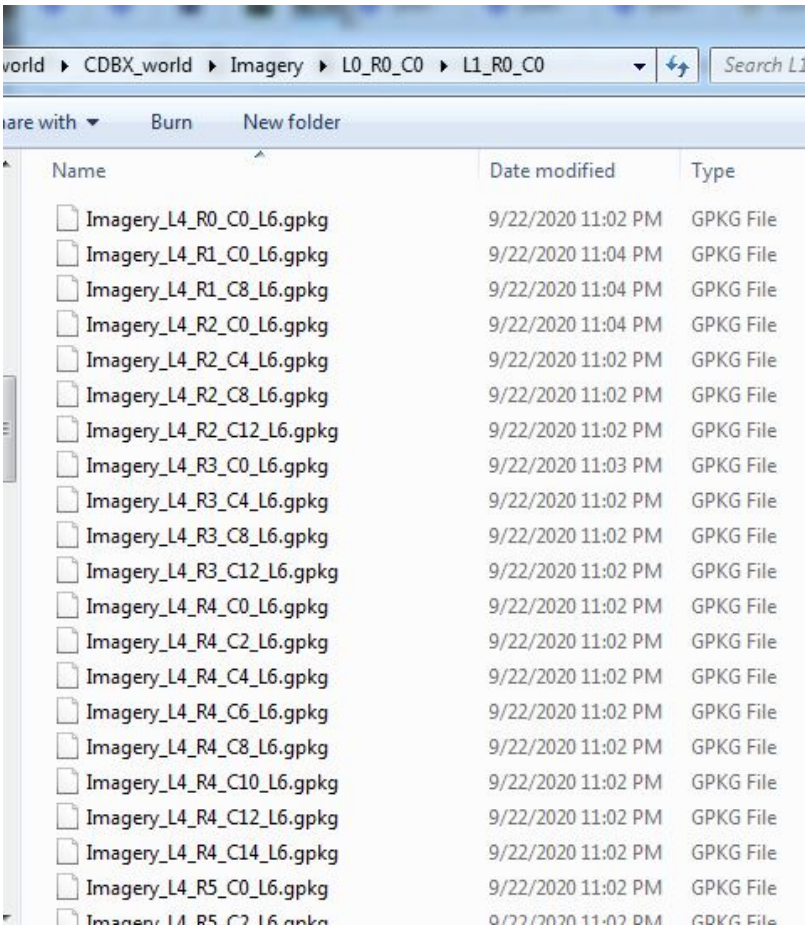


Figure Mtd_Ph3 - 44. Leaf directory structure.

Processing

This experiment uses the NASA Blue Marble imagery as world-wide imagery and USGS GTOPO30 elevation data. This provides 7 levels of detail of data (L0 to L6). Normally, the GeoPackage files should be larger for efficient use, but to show the LOD groupings, only 3 LODs are grouped together. The imagery is stored as Jpeg, so that SQLite tools can view the imagery easier, and the elevation is stored as 32-bit floating point GeoTiff files. The uncompressed data size for this experiment is around 3.05 GB.

For Experiment 3, the imagery and elevation layers were built into different GeoPackages and different directory structures. For Experiment 4, the imagery and elevation were combined into a

single set of GeoPackages and directories while keeping the LOD grouping.

Data Location

The full compressed files had to be split into multiple pieces as a multi-part zip file, to enable saving on the OGC portal.

- Experiment 3 compressed zip files containing both data layers as separate GeoPackage layers in the CDB X tiling output:
  - https://portal.ogc.org/files/?artifact_id=95374
  - https://portal.ogc.org/files/?artifact_id=95375
  - https://portal.ogc.org/files/?artifact_id=95376
- Experiment 4 compressed zip files containing both data layers in a merged GeoPackage layer in the CDB X tiling output
  - https://portal.ogc.org/files/?artifact_id=95379
  - https://portal.ogc.org/files/?artifact_id=95380
  - https://portal.ogc.org/files/?artifact_id=95381

13.15.5. Observations for Experiments 3 and 4

- The file names and directory names are pretty hard to read and understand by looking at the files. However since the tiles are rarely on a "geocell" boundary, their might not be a good naming scheme.
- Creating the LOD groupings based on the highest LOD of data makes it difficult to add data of a higher resolution later on. This might also make it harder to create "Versions" of the data that have been updated.
- There are a lot of directories created with this tiling and naming scheme. In general, there is a 1-to-1 ratio of files to directories, and directories seem to be more work for an OS to create/modify/delete.
- Current official GeoPackage standards are pretty rigid for raster data. Tiles support a very limited set of raster types (PNG or JPG), and the coverage extension supports only 16-bit PNG or 32-bit float GeoTiff. Current OGC CDB 1.1 supports data types of 8-bit unsigned, 8/16/32 bit signed, and 32-bit floating point data types, with CDB 1.2 adding the capability to support Tiff bilevel images (1-bit).
- Do we need the extra flexibility of putting different layers in different directory structures (and thus different GeoPackage files)?

CDB Versioning, old and new

Chapter 14. A review of how 'versioning' is used in legacy OGC CDB 1.X, and how that experience leads to use cases for exploration of versioning in CDB X.

Since the beginning of the CDB development, the concept of versioning was designed to support the development and distribution of small changes to the content and to support end-site CDB servers' capability to link each dataset called a CDB to a 'parent' CDB dataset using the Version.xml file in the Metadata subdirectory under the CDB-dataset top level directory.

The specific mechanism that enables versioning in OGC CDB 1.x is the XML element: <PreviousIncrementalRootDirectory name= >

The figure below, courtesy of Presagis / Hermann Brassard, shows a Version.xml file in a CDB with top-level directory named 'CDB_V1' pointing to a parent CDB top-level directory named 'CDB':

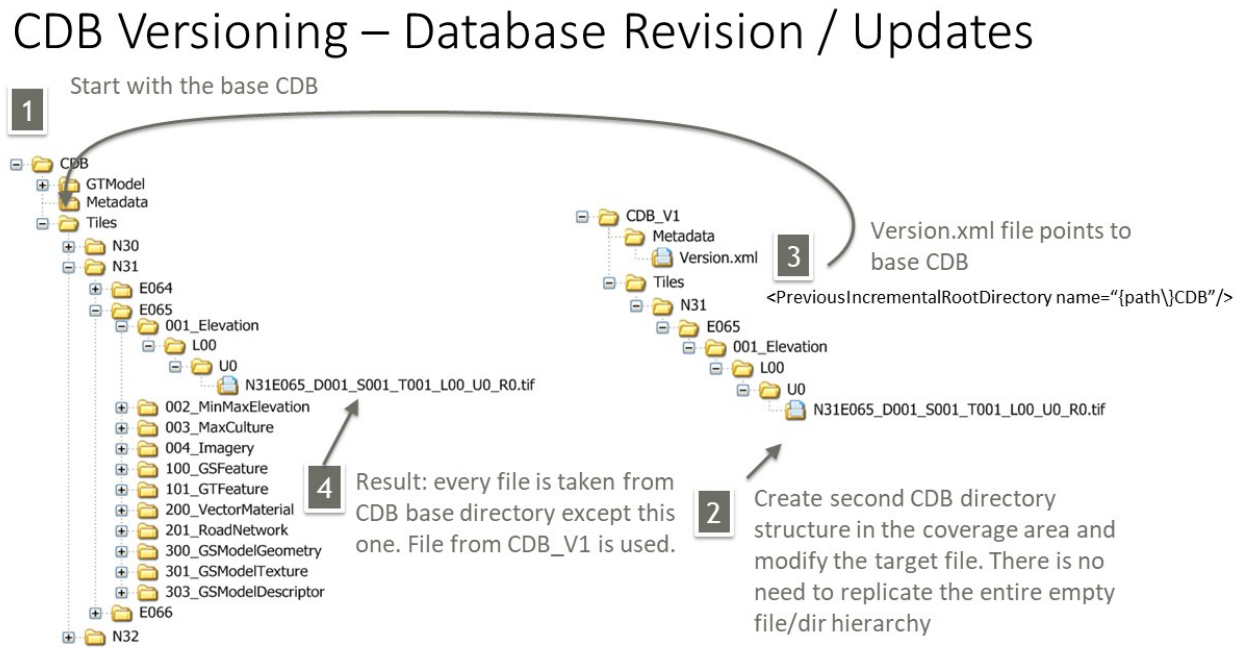


Figure Mtd_Ph3 -1. Versioning graphic: Illustrating how to use a version to 'add' a single elevation coverage file at a specific location.

OGC CDB's original versioning approach relies on simulation clients to start looking for any file at the 'bottom' of the hierarchy of CDB datasets available on the server and linked by metadata. Since simulation clients know in advance 'where they are' and 'where they want to go' a file search starts with a location. Because of the rigid directory and file naming structure of CDB, a simulation client knows where to look for any file and what the file, if present, will be named. In the example above, the simulation client would start looking in CDB_V1. In this example, CDB_V1 contains only a single file, a .tif file for N31 E065 Elevation at L00 and U0. If the simulation client is looking for anything else, it will fail to find the file it is looking for, look at Version.xml in CDB_V1, find the

<PreviousIncrementalRootDirectory name="{path}CDB"/> XML element, and look again for the file in the next higher level CDB dataset in the hierarchy.

The existing requirement is that each CDB can point to 'none' or 'one-and-only-one' parent directory, so that the search tree for the files is deterministic.

In practical use CDB servers at simulation end-sites may not be managed by experts in database architecture. CAE and other vendors will have developed utilities that allow a local or remote utility to be used at runtime client session startup to create the layout or 'chain' of CDB datasets that are present on a CDB server.

The slide below is a screen shot of a utility called the CAE CDB Update Manager Client, running on a CDB Server, that is part of a CDB simulation demonstration hosted on three laptop computers of about 2012 vintage. The first image is the laptop computer hardware, the second image is the current chain of versions on the CDB Server laptop at the time of the screenshot.

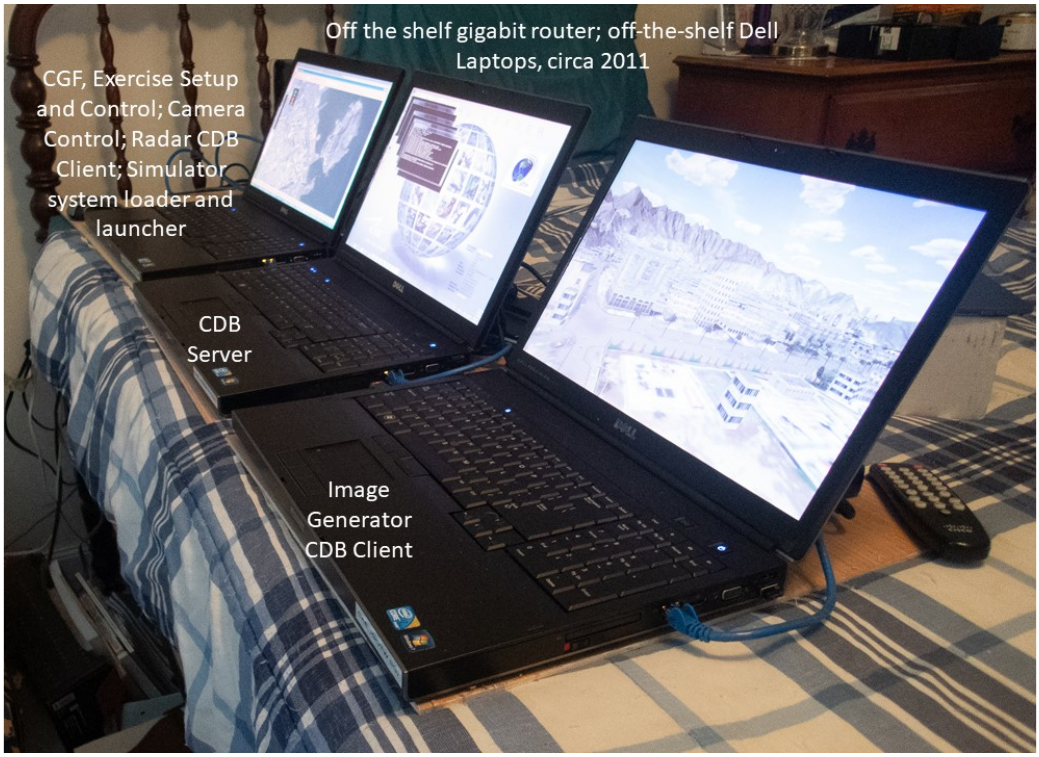


Figure Mtd_Ph3 - 45. CDB Laptop Simulation Demonstration Hardware.

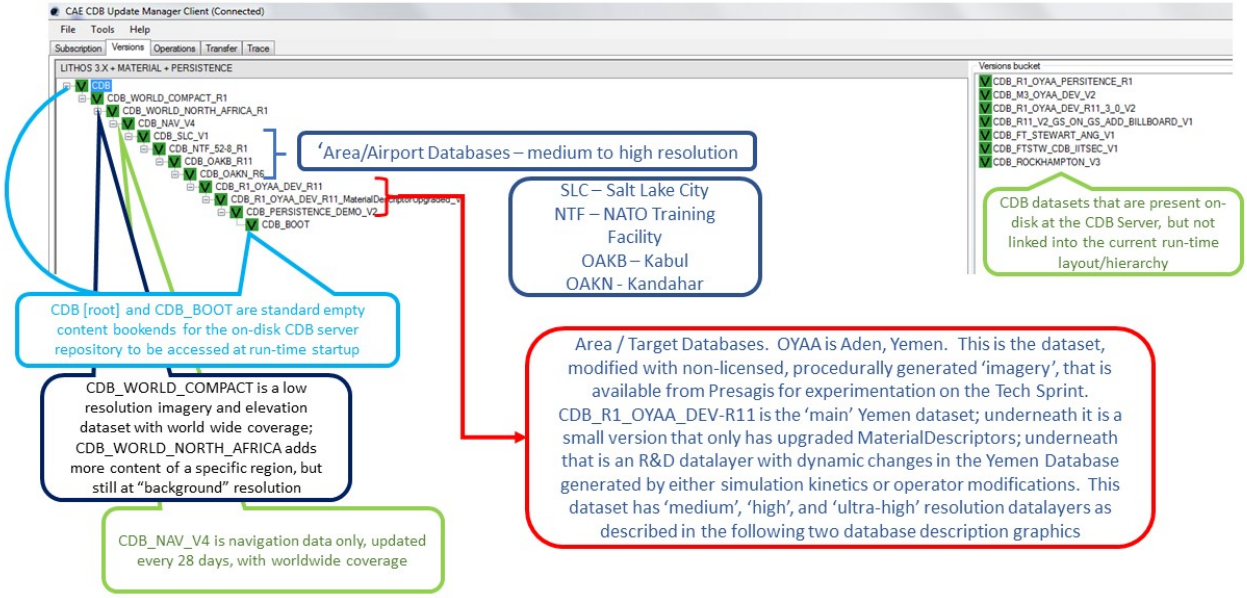


Figure Mtd_Ph3 - 46. A specific chain of CDB versions from CDB Boot at the bottom to CDB (root) at the top.

The boxes and arrows on the graphic above are the author's attempt to show just some of the flexibility available using CDB versioning. The simulator clients themselves can be set to always point to CDB_BOOT, an 'empty' CDB that only points 'up' to some content.

The top-level CDB_WORLD database can be a managed product maintained by a provider, for which regular updates are provided for example by subscription. In this example CDB_WORLD_COMPACT_R1 was created specifically for these laptop demonstrators to fit on 1TB of total HDD capability on the server laptop. At some later point some additional data for North Africa was added to blend with the subsequent development of the Yemen dataset. CDB_NAV_V4 is another product-managed dataset that provides regularly updated worldwide NAV data. There are multiple area/airport datasets that were added to the demonstration from various sources; these datasets have mutually exclusive coverage, i.e. they don't overlap in coverage. Near the bottom of this chain are the datasets from which files are found when 'location' is in the Yemen dataset coverage area. Datasets in this hierarchy can obviously be of different size, contain different LODs, could potentially contain datasets with different constraints such as security or licensing limits.

The following two graphics compare the various CDB dataset versions that comprise the Yemen data and the lower resolution ocean and world representations:

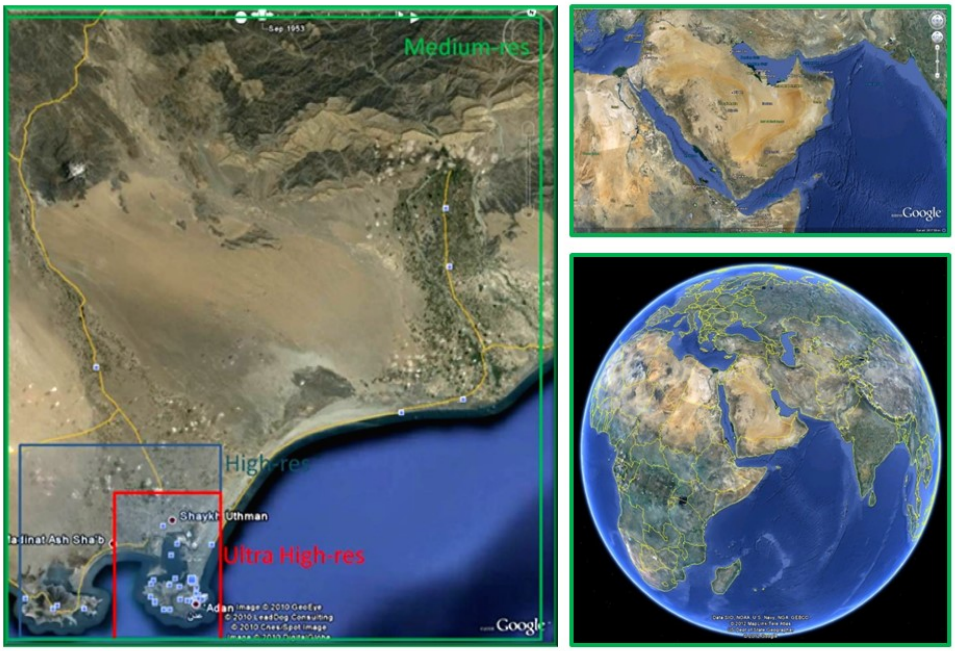


Figure Mtd_Ph3 - 47. .+

Demo Test Database Requirements					
STEP 1	Yemen Database Area				
	Ultra High-res	High-res	Medium-res	World DB	Ocean
Size	15km x 15km	20km x 20km	80 km x 80 km	500 km x 500 km	80 km x 80 km
Location	(see illustration)	(see illustration)	(see illustration)	(see illustration)	immediately south of medium-res
Elevation	2m	10m	30 m	same as World DB	same as World DB
Terrain Imagery	50 cm	50 cm	2-4 m	same as World DB	run-time generated
Light Maps	yes, buildings and roads at 1m	yes, buildings and roads at 1m	no	no	no
Raster Materials	1m	1m	1m	32m	32m
Cultural Features	fully attributed per CDB				
Buildings					
Geometry	All building footprints modeled to match terrain imagery	**Objective:** All building footprints modeled to match terrain imagery	Motif geo-typical buildings	Motif geo-typical buildings	N/A
Day/Night Textures	*Geo-typical wall imagery *Geo-specific roof texture	*Geo-typical wall imagery * Geo-typical roof imagery	Motif geo-typical buildings	Motif geo-typical buildings	N/A
Interiors	* 5 buildings with floors, walls, doors, stairs, windows,f	no	no	no	N/A
Vegetation	fully attributed per CDB				
Geometry/texture	* geo-positioned trees/bushes * At least 50 types	* Positioned by Motif * At least 50 types	Motif trees/bushes	Motif trees/bushes	N/A
Roads	fully attributed per CDB				
Geometry/texture	* Flat road surfaces * Includes medians, ditches, signage, light posts * 1 cm texture	* Flat road surfaces * Includes medians, ditches, signage, light posts * 50 cm terrain texture supplemented with road micro-texture at 1cm	* Flat road surfaces * Light points * Includes power poles/pylons * Terrain imagery	* Flat road surfaces * Light points * Terrain Imagery	N/A

Figure Mtd_Ph3 - 48. .+

So, with the above background in mind, we can look at how the CDB database(s) at an end-user site may be linked either locally or remotely by a utility or directly changing the version.xml file in each dataset so that a simulation client encounters a 'world' of data at run-time startup that can be radically different, and without any change to the basic setup and linkage between the simulation client(s) and their local CDB server.

In the following graphic, the laptop demonstration visual client is an image generator, for which the startup Viewpoint position (what I guess we should now call a GeoPose?) is 'looking' at a street

corner in Aden, Yemen. If we startup that image generator with the chain of CDB datasets shown in the 'Update Manager' window, the IG renders the scene of the street corner on the right half of the Figure:

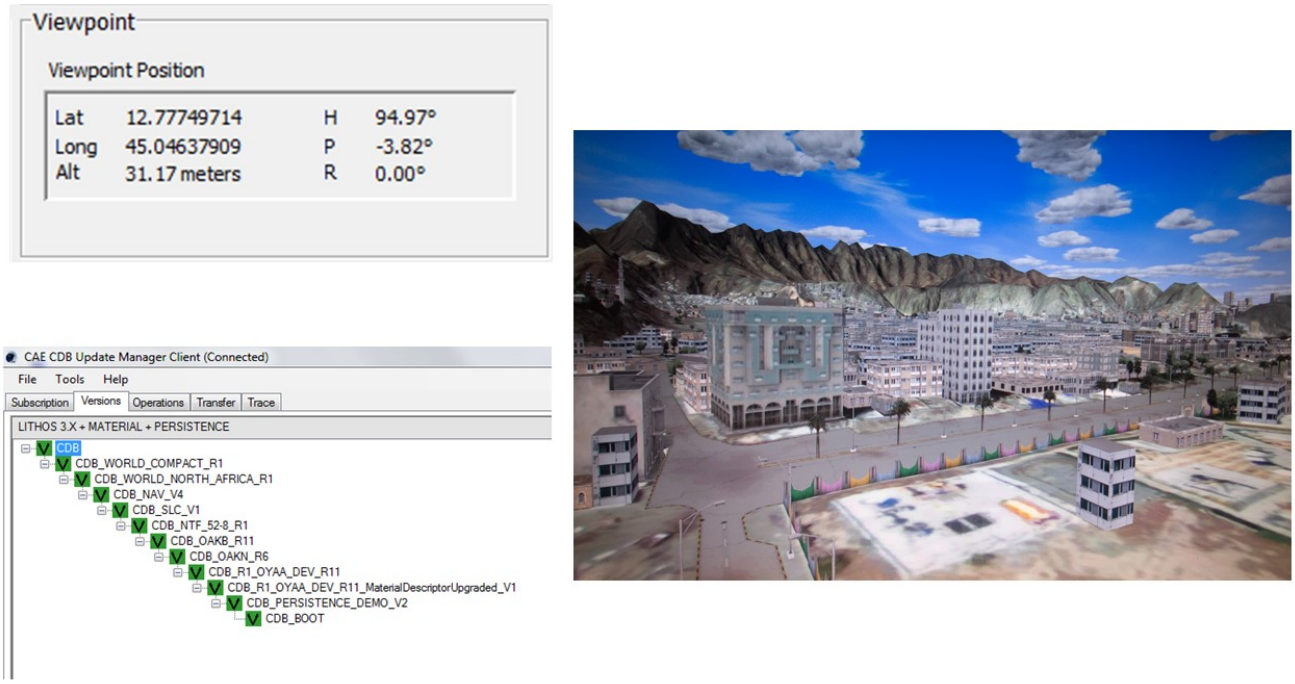


Figure Mtd_Ph3 - 49. Demonstration IG at startup with full CDB version chain from World to OYAA.+

In the following graphic, the same image generator with the same startup Viewpoint position, but with a much simpler chain of CDB datasets renders a very different scene. Note the mountains in background for comparison of the detailed Aden street view above and this view of the same location and viewpoint but with only background level imagery and elevation data:

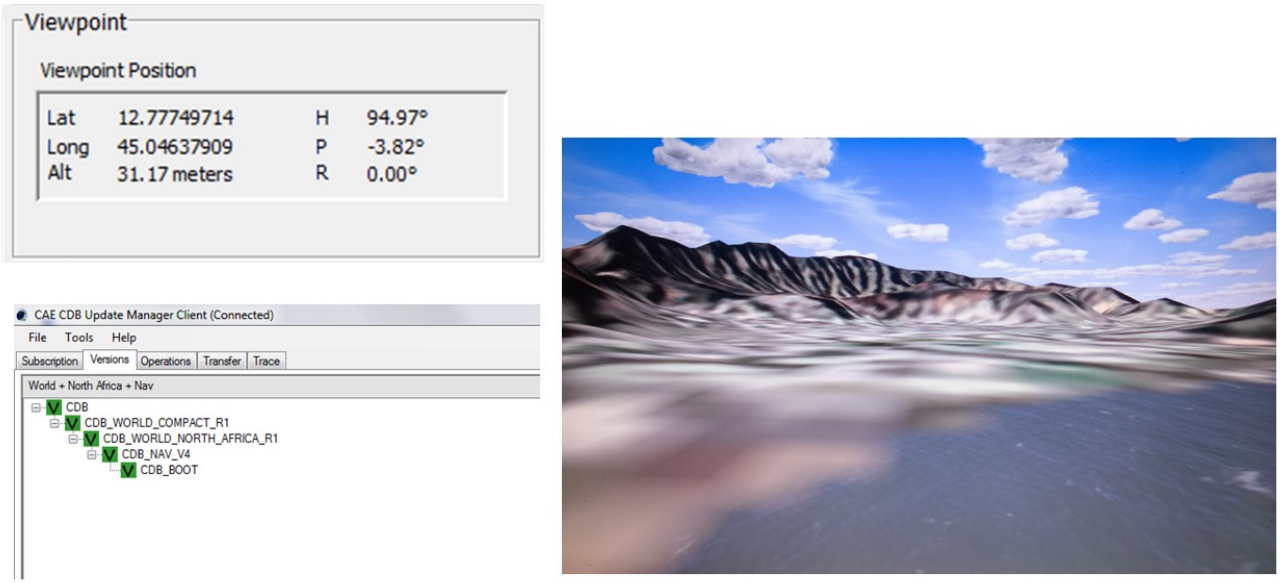


Figure Mtd_Ph3 - 50. Demonstration IG at startup with full CDB version chain from World to OYAA.+

14.1. Summary

Versioning is used extensively in fielded CDB implementations. A primary principal of the current approach is the direct replacement of a file in a parent CDB datasate by an identically named file in a linked child dataset. Anyone at any level in the operation of the site of the simulators is able to assume the files are 'packaged' identically in every CDB dataset, whether it was produced locally, or produced as a product purchased, or acquired through sharing, and so forth.

Versioning is used to distribute a small change very rapidly. In a previous Integration Analysis document, this author asked all those experimenting in this Tech Sprint to declare how their implementation would deal with a change in which a bridge near this street corner in Yemen is damaged/destroyed. In CDB 1.X, that small change could be a single Openflight file in a CDB version. This very small CDB version can be very quickly developed and transported to widely dispersed simulation sites. Including the change in the 'world' of end site simulation clients after the CDB dataset containing the changed file is on the local CDB server requires rewriting only two version.xml files and restarting the simulation clients. In actual practice, if the original Openflight model included one or more damage states, the change could actually be accomplished by a selector for the damaged model instead of the default/undamaged model.

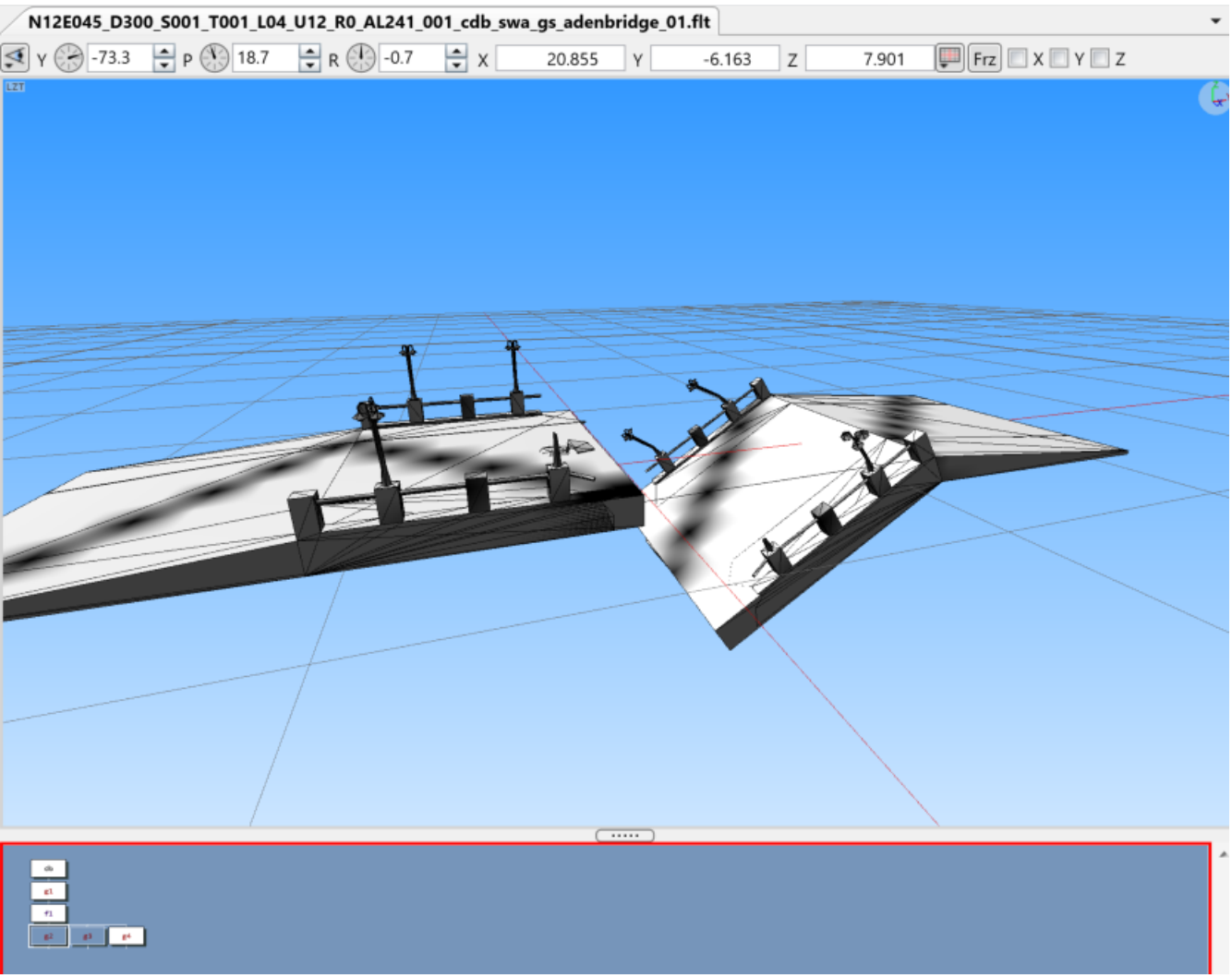


Figure Mtd_Ph3 - 51. Openflight damage 'state' of Sira Island Bridge.+

Versioning is used so that background level data, including things like worldwide NAV data, can be maintained and distributed on a different schedule than other more localized data.

Versioning is used so that CDB datasets that may have some constraint in one or more layers (such as licensing or security classification) can be distributed separately from unrestricted datasets of the same coverage, and linked into the version chain when appropriate.

Versioning is used to proved temporary changes to a dataset, such as fortifications or obstacles or other objects that support a specific training or rehearsal, but are not part of the permanent CDB collection for the same coverage.

Versioning is also used to quickly disseminate changes that will become part of the permanent collection, but are able to be linked immediately for use and 'flattened' into a parent coverage when appropriate at a later date.

==The German Army Aviation Training School in Buchenberg, Germany, has 12 helicopter flight simulators (CH-53, UH-1D, EC-135, with multiple CDB clients (out-the-window visuals; non-visual sensors; computer-generated-forces); there are other simulators within the German Armed Forces that also have CDB simulator clients and that share interoperable CDB datasets with the Army helicopter school.

The CDB holdings include world-wide background data; higher resolution datasets for Europe and Germany; some country data procedurally generated; some airports procedurally generated; some country data created from satellite imagery and hand modelling; some airports created from satellite imagery and hand-modelling; both the procedurally-generated and satellite/hand modelled data under continuous update and generation of additional content

There are multiple distributed CDB servers; ther is a main 'enterprise' facility CDB repository on site, there are offsite CDB repositories and content creation facilities at the Luftwaffe Fighter Weapons Center Synthet Environment site and the CAE Stollberg.

The CDB versioning capability is integral to management of the site.

Appendix A: Annex A: Summary of Recommendations

Chapter 15. Attribution Subgroup

The Attribution Subgroup recommends extending the current CDB XML metadata to add the NAS metamodel capabilities that are currently not supported. The CDB core conceptual model should not mandate any particular data dictionary or content, but rather provide the conceptual and logical metamodel for describing any ISO 19109 compliant application schema to the maximum extent practical. There should be no technical reason why one could not develop an extension profile for CDB for any particular data dictionary that complies with ISO 19109.

- Adopt NAS-compliant logical entity-attribute model for CDB X with extensions for CDB use cases
  - Store all aspsects of feature and attribute dictionary in single SQLite file for portability and runtime performance
  - Use GGDM 3.0 as the initial starting point for definitions to populate CDB X data dictionary
  - Match up remaining CDB non-GGDM feature types and subcodes with latest NAS definitions where possible, matching subcodes to attributes where relevant and adding missing enumerant values where necessary (with associated NCV vocabulary)
  - Augment NAS definitions with other open standards and new development
    - Match up missing maritime definitions to DGIM followed by IHO S-100 where possible, define NCV vocabulary for such integrated definitions
    - Match up missing aeronautics definitions to DGIM followed by AIXM where possible, define NCV vocabulary for such integrated definitions
    - Coordinate with OGC, NGA, and STE OWT to develop replacement building interior data model incorporating IFC/BIM, CityGML, IMDF, and other open standards
    - Coordinate with OGC, NGA, STE OWT, and civilian environment agencies to develop detailed data model for vegetation
    - Create data model and vocabulary for material and light point definitions and capture into existing material and lightpoint libraries
    - Define NCV vocabulary and NAS-compatible entity and attribute types for CDB feature types and subcodes totally missing in all other standards
  - Remove CDB feature subcodes entirely; migrate to existing and new feature and attribute types instead in NAS-compliant structure
  - Define entity types for CDB datasets and define "aggregation" relationships from feature types to containing datasets
  - Capture feature-level, dataset-level, and database metadata as NAS-compliant attribution meeting the NSG Metadata Foundation (NMF) and ISO 19115
  - Define functional role domains and create mechanism to organize attribution by domain for tailoring to runtime devices
- Delegate entity and attribute physical encoding choices to vector and 3D model containers instead of specifying globally
  - Deprecate extended attributes entirely, to be removed in CDB X
  - Delegate containerization of entity types (one per table, multiple in same table specified by

F_CODE attribute, etc.) to vector and model containers

- Delegate decision whether to use class or instance attributes to individual vector and model containers rather than global data dictionary
- Delegate decision of whether to use FACC-like codes, NAS labels, or natural language names for entity types, attributes, and values to vector and model containers
- Delegate decision of whether to flatten complex feature relationships and attributes used GGDM attribute prefixing to vector and model containers
- Delegate decision of whether to flatten multi-valued and range-valued attributes using GGDM attribute prefixing to vector and model containers
- Specify minimum and maximum cardinality of multi-valued attributes in feature and attribute dictionary, allow containers to use a lower maximum if using GGDM attribute prefixing encoding

- Define backward-compatible extensions in CDB 1.3 to add constructs necessary to move toward NAS-compliant attribution
  - Capture proposed NAS-complaint replacement feature dictionary in existing CDB metadata XML with necessary extensions
  - Only use feature subcode 000 in replacement dictionary and deprecate use of feature subcodes to be removed in CDB X
  - Add mechanism to mark numeric attributes as interval ranges (existing non-upgraded clients should see still attribute as single-valued and read mean value from associated unsuffixed attribute, use suffixed attributes for deviation and closure for upgraded clients to read)
  - Add minumum and maximum cardinality elements for attribute definitions to specify mininum and maximum element count for multi-valued attributes (existing non-upgraded clients should just see attribute as scalar base value and will only read the first value from associated content, will see ordinal-suffixed attributes as separate attributes)
  - Add list of valid attributes to datasets in CDB 1.x metadata XML files to match existing human-readable specification
  - Add list of valid enumerants for each attribute in CDB 1.x CDB_Attributes.xml file to match existing human-readable specification
  - Add list of valid attributes for each entity type as extension to CDB 1.x Feature_Data_Dictionary.xml to implement NAS-compliant per-entity attributes
  - Update CDB 1.x CDB_Attributes.xml to allow specifying text pattern constraints through <Pattern> element and text codelists for text attributes via <Codelist> element
  - Update CDB 1.x Feature_Data_Dictionary.xml for each feature to specify its generalization (base) entity type via <Generalization> element
  - Update CDB 1.x Feature_Data_Dictionary.xml to add <Aggregation> element to define additional associated category for an entity type, or parent category for a category
  - Existing category and subcategory XML structure will add implicit definitions and aggeregation links for the category/subcategory items as used by CDB 1.0 for model storage

`Recommendation 1 Extended Attributes - Version 1.3` The subgroup discussion on this topic is titled: Should Extended Attributes be preserved at the logical data model level? [https://github.com/sofwerx/cdb2-concept/issues/25] The suggestion is that the CDB SWG discuss this issue and possible solution as a possible change for CDB version 1.3. Some additional testing may be required to determine if this capability can be added to version 1.3 or not.

`Recommendation 2 Attribute default values - Version 1.3` The subgroup discussion on this topic is titled: Attribute Default Values #32 [https://github.com/sofwerx/cdb2-concept/issues/32]. The recommendation is that Defaults.xml can be used to define global attribute defaults as well as per-dataset defaults. Doing per-entity defaults would be a straight forward extension that could be proposed for CDB 1.3 as a transition path. The subgroup suggests that the CDB SWG discussion this for possible inclusion in version 1.3. A change request for this approach to specifying default values is also suggested.

`Recommendation 3 Attribute Terms - Version 1.3` The subgroup discussion on this topic is titled: Capture Attribute Terms (Enumerants) in Metadata #31 [https://github.com/sofwerx/cdb2-concept/issues/31]. Attributes describing qualitative values are present in CDB 1.2 and the list of valid values for each attribute are documented in the human-readable specification with both the vocabulary term name and its integer numeric value (index). However, the machine-readable XML metadata does not contain any of this information and treats these attribute types as raw integers with only a minimum and maximum value constraint. It may make sense as a transition path to update CDB 1.3 to define additional XML elements in a backward compatible way to capture these definitions from the existing specification into the machine-readable XML metadata. The conceptual model in the CDB 1.2 specification does align with how GGDM treats such attributes, so there is no fundamental incompatibility, and the proposed CDB X dictionary design accounts for properly tracking the terms for qualitative attributes in a machine-readable way in SQLite.

Appendix B: Annex B: CDB-X sample GeoPackages resulting from Ecere experiments.

The results produced were sourced from the CAE San Diego CDB 1.x.
An additional data layer was sourced from the NASA Visible Earth Blue Marble [https://visibleearth.nasa.gov/collection/1484/blue-marble].
A detailed description of these results is featured in the Engineering Report – Tiling section. [https://github.com/sofwerx/cdb2-eng-report/blob/master/11-tiling-coverages.adoc]
These are also available from the CDB SWG / CDB X Tech Sprint folder [https://portal.ogc.org/index.php?m=projects&a=view&project_id=466&tab=2&artifact_id=95315].

MVT: Mapbox Vector Tiles (specifications [https://docs.mapbox.com/vector-tiles/reference/])

GMT: GNOSIS Map Tiles (specifications [http://docs.opengeospatial.org/per/18-025.html#GMTSpecs])

E3D: Ecere 3D Models (specifications [http://docs.opengeospatial.org/per/18-025.html#E3DSpecs])

Chapter 16. San Diego CDB layers packaged together

Data / Link	LOD Grouping	Models Encoding	Imagery Encoding	Coverage Encoding	Vector Encoding	Uncompressed size	Compressed Size
As a single GeoPackage							
San Diego [https://data.ogc.org/2020/11/SanDiego.gpkg.7z]	Single File	Binary glTF	JPEG	PNG	MVT	12.8 GiB	9.8 GiB
San Diego [https://data.ogc.org/2020/11/SanDiego.gpkg]	Single File	E3D	JPEG	GMT	GMT	10.3 GiB	
Grouping tiles of 5 LODs per GeoPackage							
San Diego [https://data.ogc.org/2020/11/SanDiego.7z]	5	Binary glTF	JPEG	PNG	MVT	14.6 GiB	9.9 GiB
San Diego (subset) [https://portal.ogc.org/files/?artifact_id=95370]	5	Binary glTF	JPEG	PNG	MVT	1.4 GiB	217.5 MiB

Chapter 17. San Diego CDB packaged as separate layers

17.1. Elevation

Data / Link	LOD Grouping	Coverage Encoding	Uncompressed size	Compressed Size
Elevation [https://portal.ogc.org/files/?artifact_id=95352]	6	PNG / 16-bit integer	483 MiB	472 MiB
Elevation [https://portal.ogc.org/files/?artifact_id=95328]	6	GMT / 16-bit integer	370.8 MiB	365.2 MiB
Elevation [https://data.ogc.org/2020/11/ElevationTIF.7z]	6	GeoTIFF / 32-bit float	1.4 GiB	1.3 GiB

17.2. Imagery

Data / Link	LOD Grouping	Imagery Encoding	Uncompressed size	Compressed Size
Blue Marble [https://data.ogc.org/2020/11/BlueMarble.7z]	7	JPEG	1.6 GiB	1.4 GiB
Imagery (Medium) [https://data.ogc.org/2020/11/ImageryMedium.7z]	7	JPEG	4.9 GiB	4.7 GiB
Imagery (High) [https://data.ogc.org/2020/11/ImageryHigh.7z]	7	JPEG	4.5 GiB	4.3 GiB

17.3. Vector data

Data / Link	LOD Grouping	Vector Encoding	Uncompressed size	Compressed Size

Hydrography [https://portal.ogc.org/files/?artifact_id=95348]	Single File	Mapbox Vector Tiles	424 KiB	
Roads [https://portal.ogc.org/files/?artifact_id=95350]	Single File	Mapbox Vector Tiles	69.1 MiB	17.6 MiB
Airport Lights [https://portal.ogc.org/files/?artifact_id=95346]	Single File	Mapbox Vector Tiles	144 KiB	
Hydrography [https://portal.ogc.org/files/?artifact_id=95330]	Single File	GNOSIS Map Tiles	248 KiB	
Roads [https://portal.ogc.org/files/?artifact_id=95331]	Single File	GNOSIS Map Tiles	38 MiB	23.4 MiB
Airport Lights [https://portal.ogc.org/files/?artifact_id=95329]	Single File	GNOSIS Map Tiles	148 KiB	

17.4. 3D Models

17.4.1. Using referencing placement points

Data / Link	LOD Grouping	Models Encoding	Vector Encoding	Uncompressed size	Compressed Size
Buildings [https://portal.ogc.org/files/?artifact_id=95351]	Single File	Binary glTF	Mapbox Vector Tiles	2.9 GiB	321.3 MiB
Trees [https://portal.ogc.org/files/?artifact_id=95349]	Single File	Binary glTF	Mapbox Vector Tiles	2 MiB	
Coronado Bridge [https://portal.ogc.org/files/?artifact_id=95347]	Single File	Binary glTF	Mapbox Vector Tiles	272 KiB	

Buildings [https://portal.ogc.org/files/?artifact_id=95340]	Single File	E3D	GNOSIS Map Tiles	560.8 MiB	332.8 MiB
Trees [https://portal.ogc.org/files/?artifact_id=95339]	Single File	E3D	GNOSIS Map Tiles	1.8 MiB	
Coronado Bridge [https://portal.ogc.org/files/?artifact_id=95338]	Single File	E3D	GNOSIS Map Tiles	196 KiB	
Buildings [https://portal.ogc.org/files/?artifact_id=95344]	4	Binary glTF	Mapbox Vector Tiles	4.1 GiB	446.6 MiB
Buildings [https://portal.ogc.org/files/?artifact_id=95343]	5	Binary glTF	Mapbox Vector Tiles	3.7 GiB	409.2 MiB
Buildings [https://portal.ogc.org/files/?artifact_id=95342]	7	Binary glTF	Mapbox Vector Tiles	2.9 GiB	321.3 MiB
Trees [https://portal.ogc.org/files/?artifact_id=95341]	6	Binary glTF	Mapbox Vector Tiles	2.9 MiB	1.2 MiB

17.4.2. Batching models per tile

Data / Link	LOD Grouping	Models Encoding	Vector Encoding	Uncompressed size	Compressed Size
Buildings [https://portal.ogc.org/files/?artifact_id=95337]	Single File	Binary glTF	Mapbox Vector Tiles	3.9 GiB	512.4 MiB
Trees [https://portal.ogc.org/files/?artifact_id=95335]	Single File	Binary glTF	Mapbox Vector Tiles	91 MiB	2.3 MiB

Coronado Bridge [https://portal.ogc.org/files/?artifact_id=95336]	Single File	Binary glTF	Mapbox Vector Tiles	576 KiB	
Buildings [https://portal.ogc.org/files/?artifact_id=95334]	Single File	E3D	GNOSIS Map Tiles	417.3 MiB	339.8 MiB
Trees [https://portal.ogc.org/files/?artifact_id=95333]	Single File	E3D	GNOSIS Map Tiles	3.3 MiB	
Coronado Bridge [https://portal.ogc.org/files/?artifact_id=95332]	Single File	E3D	GNOSIS Map Tiles	256 KiB	

For the GeoPackage CDB Interoperabilty Experiment, FlightSafety strictly tested existing Shapefile performance vs GeoPackage performance from a flight simulation runtime perspective. So how fast can we pull data out of a GeoPackage in the same manner that we process shapefiles. GeoPackage creation timing wasn't that interesting to us at the time. We were also testing the different approaches put forward on how to structure the GeoPackage to contain the data, whether that was one GeoPackage containing a single shapefile's features, one GeoPackage table containing all the features at a given level of detail, or one GeoPackage table containing all the features at all the levels of detail. Changing the CDB conceptual model or levels of detail was outside the scope of this project.

FlightSafety found that querying large numbers of features out of very large tables was dependent on how the GeoPackage was structured. The original conversion of the field columns queried was to a string, making these fields integers made it faster, but creating an index on those columns was an order of magnitude faster. The table here shows the three different GeoPackage organizations, how many features ended up in the table we are querying, and how fast we could pull out 16k of rows from that table. Each test pulled out the same set of 16k rows. The highlighted data shows how structuring the GeoPackage matters when it comes to performance.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2						SQL Query on String Fields			SQL Query on Integer Fields			SQL Query on Indexed Integers	
3													
4	Test Number	Features Returned in Query	Total Features in Table	% Returned		Per Test (ms)	Normalized		Per Test (ms)	Normalized		Per Test (ms)	Normalized to Opt 1
5	1	16,384	16,384	100.00%		96.60	**1.00**		87.34	**1.00**			
7	2	16,384	3,375,935	0.49%		1125.67	**11.65**		1330.79	**15.24**		138.17	**0.63**
8	3	16,384	6,865,325	0.24%		3025.29	**31.32**		2677.78	**30.66**		173.90	**0.50**
9													
10	All features contain the row and column and lod from the CDB tile they were within before the conversion to GeoPackage												
11	Timing was averaged over multiple runs, and includes the I/O of opening and closing the GeoPackage												
12													
13	First Test Query:		**SELECT * FROM table_name**										
14	Second and Third Test Query:		**SELECT * FROM table_name WHERE ROW = x AND COL = y AND LOD = l**										
15													
16	Index for the last timing column created with:			**CREATE INDEX**	**'101_GTFeature_S002_T001_idx' on '101_GTFeature_S002_T001' (LOD, ROW, COL)**								
17													

Figure Mtd_Ph3 - 52. Performance.

Annex C: Revision History

replace below entries as needed

Table 1. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
June 2020	C. Reed	.1	all	Clone template and initial version
November 2020	Graham and Reed	1.0	All	Complete final draft.

Annex D: Bibliography

[1] STAC: Spatio Temporal Asset Catalogs. [https://stacspec.org/] (2017).

[2] Ground-Warfighter Geospatial Data Model (GGDM) [https://ggdm.erdc.dren.mil/] (2020)