OGC Testbed-13

CDB Engineering Report

Table of Contents

Publication Date: 2018-01-11

Approval Date: 2017-12-07

Posted Date: 2017-11-15

Reference number of this document: OGC 17-042

Reference URL for this document: http://www.opengis.net/doc/PER/t13-NG001

Category: Public Engineering Report

Editor: Sara Saeedi, other participants of the CDB thread

Title: OGC Testbed-13: CDB Engineering Report

OGC Engineering Report

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. Summary

The Open Geospatial Consortium (OGC) CDB Standard is an international standard that describes rules, requirements, and best practices for structuring, modeling, and storing geospatial information required in high performance modeling and simulation (M&S) applications. Previously called the Common Data Base Specification the OGC members voted to officially change the name to “CDB” in the OGC community and standard [4]. OGC CDB 1.0 defines the core conceptual models, use cases, requirements, and specifications for employing geospatial data in 3D M&S. A persistent and important concern about the OGC CDB is how to evolve beyond the current use of CDB feature codes and incorporate the more recent NSG (National System for Geospatial-Intelligence) Application Schema (NAS) and DGIWG (Digital Geographic Information Working Group) Feature and Attribute encoding. The CDB Standard Working Group (SWG) has started examining the feature codes and attribution schema as used in the CDB standard. This effort was expanded in Testbed 13 to develop CDB profiles for supporting interoperability of existing CDB with the NAS-based feature codes and schemas by investigating how a specific profile, *e.g.* "Urban Military NAS Profile" can be developed and used in CDB. For this purpose, a mapping between CDB and NAS feature codes and schemas is expressed. This mapping has been applied by OGC Web Services such as Web Feature Service (WFS) to publish a CDB feature with attributes and schemas from the NAS profile.

This Engineering Report (ER) summarizes the CDB sub-thread work in Testbed 13. The document is structured in three phases and includes 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 Engineering Report describes:

- 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).

This work provides insights into:

- The in-depth study of the OGC CDB 1.0 feature data dictionary and attribution schema;
- The requirements and constraints for extending the CDB feature data dictionary (FDD) and attribute schemas;
- The development and prototyping of the WFS and WCS access to the CDB datastore for a NAS based urban military scenario.

1.1. Requirements

This Engineering Report addresses the following requirements:

- A NAS-based profile UML model to configure the feature data in a CDB dataset;

- Creation of the CDB configuration mapping files consistent with a NAS profile UML model;
- Implementation of a CDB dataset with all the requirements of a NAS-based Urban Military profile.

1.2. Key Findings and Prior-After Comparison

The work and experiments completed in the OGC Testbed 13 CDB sub-thread are described in this CDB ER document. This ER was reported to the CDB SWG on several occasions and the current key findings of the CDB sub-thread are summarized here:

- Feasibility study for extending OGC CDB 1.0 feature codes and attributes reveals that the current version (1.0) supports only a limited extension:
    - The feature type codes which exist in the FDD.xml files are a fixed set, and the extension needs hard coding of the newly added codes;
    - CDB has an indexing structure (based on Feature and Attribute Coding Catalogue (FACC) codes) to save feature types inside the folder hierarchy. This indexing is also a limitation for defining the feature codes. If an arbitrary alphabetical feature code is used inside the CDB, then the indexing mechanism will be broken. Another approach is to define a mapping between the new alphabetical feature codes and old indexing;
    - The attributes schema is defined for the tiled vector datasets inside the Shapefile database file. However, the attribute schema can be extended in the current version of the CDB using extended attributes.
- The OGC CDB 1.0 can be accessed by OGC web services that support a NAS profile in organizing features and attributes and indexing vector features.
- Change Request proposals have been discussed for the OGC CDB 2.0 and will be submitted by December 31, 2017.
    - The 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;
    - The recommendations for supporting application schemas in CDB (level of complexity: Esri Geodatabase, GML-SF0 application schema) are being discussed for the OGC CDB 2.0;
    - The method to expand the supported encodings and formats for an OGC CDB compliant datastore;
    - Generating a coherent attribute schema for CDB 1.0 based on the "CDB_Attribute.xml" file.

1.3. What Does This ER Mean for the Working Group and OGC in General

This ER is mainly of interest to the CDB SWG and the CDB implementation community. In particular, any Change Request Proposals created as a result of the work will be discussed in the SWG. The work documented in this ER is important for CDB SWG as it applies to the CDB 1.0 standard, expands its feature codes/attribution schema using NAS profile, and accesses the CDB data store using OGC web services. Due to the strong link to 3D information streaming and NGA's

NSG application schema, the work is also expected to be of interest to other OGC Domain Working Groups (DWG) such as the 3D Information Management (3DIM) DWG, Defense & Intelligence DWG, Distributed Simulation and Gaming DWG, OGC Web Services and profiling.

1.4. Document Contributor Contact Points

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

Table 1. Contacts

Name	Organization
Sara Saeedi (editor and Contributor)	University of Calgary
Steve Liang (Contributor)	University of Calgary
Robin Houtmeyers (Contributor)	Luciad Inc.
James Badger (Contributor)	University of Calgary

1.5. Future Work

The following future work items are envisioned:

1. A performance analysis and evaluation for using CDB extended with the NAS profile in run-time simulation;
2. Describe explicitly how the CDB model may or may not align with the rest of the OGC baseline, including standards such as Sensor Observation Service (SOS) and the Discrete Global Grid Systems (DGGS) standard;
3. Extend the supported encodings and formats for a CDB database to include the use of the OGC GeoPackage, CityGML, LandInfra and IndoorGML standards as well as other broadly used community encoding standards, such as COLLADA. This work may require performing OGC interoperability experiments to understand the implications of these options better.
4. Consider replacement(s) for the feature data dictionary with application schemas (*e.g.*, NAS) that describe the features, attributes, associations and relationships of a different domain of application.

1.6. Foreword

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

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

Chapter 2. References

The following normative standards are referenced in this document.

- ISO 19109:2005, Geographic information — Rules for application schema, June 2005 [https://www.iso.org/standard/39891.html]
- ISO/IEC 18025:2014, Information technology — Environmental Data Coding Specification (EDCS), February 2014 [https://www.iso.org/standard/60505.html]
- OGC 15-112r2, OGC® CDB Core Standard: CDB Terms and Definitions, February 2017 [https://portal.opengeospatial.org/files/?artifact_id=72714]
- OGC 15-113r3, OGC® CDB Core Standard: Model and Physical Data Store Structure Standard, February 2017 [https://portal.opengeospatial.org/files/?artifact_id=72712]
- OGC 16-007r3, OGC® CDB Core Standard Conceptual Model Standard, February 2017 [https://portal.opengeospatial.org/files/?artifact_id=72723]
- OGC 06-042, OpenGIS Web Map Service (WMS) Implementation Specification, March 2006 [http://portal.opengeospatial.org/files/?artifact_id=14416]
- OGC 09-110r4, OGC® WCS 2.0 Interface Standard - Core, version 2.0.1, July 2012 [https://portal.opengeospatial.org/files/09-110r4]
- OGC 09-025r2, OGC® Web Feature Service 2.0 Interface Standard – With Corrigendum, July 2014 [http://docs.opengeospatial.org/is/09-025r2/09-025r2.html]
- OGC 07-036, OpenGIS Geography Markup Language (GML) Encoding Standard, August 2007 [http://portal.opengeospatial.org/files/?artifact_id=20509]

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

Chapter 3. Terms and definitions

For the purposes of this report, the definitions specified in the OGC® CDB Core Standard: CDB Terms and Definitions (OGC 15-112r2 [https://portal.opengeospatial.org/files/?artifact_id=72714]) shall apply. In addition, the following terms and definitions apply.

3.1. Application Schema

```
An application level conceptual schema for data required by one or more applications.
It includes feature type and model (code, attributes, units, etc.) along with any
constraints important for the integrity of the data.
```

3.2. Client

```
The software component that can invoke an operation from a service.
```

3.3. Feature

```
A fundamental unit of spatial data that represents a physical entity, e.g. a building,
a river, or a person. A feature may or may not have geometric aspects.
```

3.4. 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.
```

3.5. GeoServer

```
An open source map server for sharing geospatial data designed for interoperability.
```

3.6. GSModel

```
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.
```

3.7. GTModel

```
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.
```

3.8. Interface

```
The named set of operations that characterize the behaviour of a service.
```

3.9. OpenFlight

```
OpenFlight (or .flt) is a 3D geometry model file format designed as a non-proprietary
format for use by real-time 3D visual simulation image generators.
```

3.10. Service

```
The distinct part of the functionality that is provided by an entity through
interfaces.
```

3.11. QGIS

```
A Free and Open Source Geographic Information System http://qgis.org[QGIS Website]
```

Chapter 4. Abbreviated terms

- 3D Three Dimensional
- 3DIM 3D Information Management
- API Application Program Interface
- EA Enterprise Architect (UML modeling tool from Sparx Systems)
- ER Engineering Report
- Esri Environmental Systems Research Institute
- ETag Entity tag
- CDB Common Database (OGC standard)
- CEAI CDB Extended Attribute Index
- COLLADA COLLAborative Design Activity
- COTS Commercial Off The Shelf
- CRS Coordinate Reference System
- DFDD DGIWG Feature Data Dictionary
- DGIWG Defense Geographic Information Working Group
- DIGEST The Digital Geographic Information Exchange Standard
- DIS Distributed Interactive Simulations [IEEE Std 1278TM]
- DWG Domain Working Group
- EAC Environment Attribute Code
- EAV Environment Attribute Value
- FACC Feature and Attribute Coding Catalogue
- FDD Feature Data Dictionary
- FSC Feature Sub-Code
- GEAI Geomatics Extended Attribute Index
- glTF graphics library Transmission Format
- GML Geography Markup Language (OGC/ISO standard)
- GML-SF GML Simple Features Profile (OGC standard)
- GML-SF0 GML Simple Features Profile, Level 0
- GPU Graphics Processing Unit
- HLA High-Level Architecture Standard [IEEE Std 1516TM]
- HTML Hypertext Markup Language
- HTTP Hypertext Transfer Protocol
- ISO International Standardisation Organisation
- JPEG Joint Photographic Experts Group

- KVP Key-Value Pair
- LoD Level of Detail
- LZW Lempel–Ziv–Welch method for image compression
- M&S Modeling and Simulation
- MModel Moving 3D models
- NAS NSG Application Schema
- NetCDF Network Common Data Form
- NGA National Geospatial-Intelligence Agency
- NSG National System for Geospatial-Intelligence
- NYC New York City
- OGC Open Geospatial Consortium, Inc.
- TIE Technology Integration Experiment
- TIFF Tagged Image File Format.
- SGI Silicon Graphics Image
- SWG Standard Working Group
- UML Unified Modeling Language (OMG standard)
- URL Uniform Resource Locator
- VEAI Vendor Extended Attribute Index
- WebGL Web Graphics Library
- WCS Web Coverage Service (OGC standard)
- WFS Web Feature Service (OGC standard)
- WGS-84 World Geodetic System 1984
- WMS Web Map Service (OGC standard)
- WMTS Web Map Tile Service
- XML Extensible Markup Language (W3C standard)
- XSD XML Schema Definition

Chapter 5. Overview

This Engineering Report (ER) discusses the conduct and results of the CDB extended future codes and attribute schemas Feasibility Study, including all lessons learned from the experiments completed during the CDB feature code transformation development and the web services implementation. The purpose of this document is to summarize the project definition, requirements, and evaluate proposed solutions in order to offer the optimum recommendations. The CDB ER will explain the current CDB data model, attribute schema as well as the feasibility of expanding the current schema. The web service and client-server architecture for the WFS, WCS, and conversion client for accessing the CDB data store is presented in the ER. Then, the results of the experiments for online performance are evaluated. Finally, the project recommendations are presented.

5.1. Outline

The following figure (Figure 1) describes the CDB workflow executed in Testbed 13.

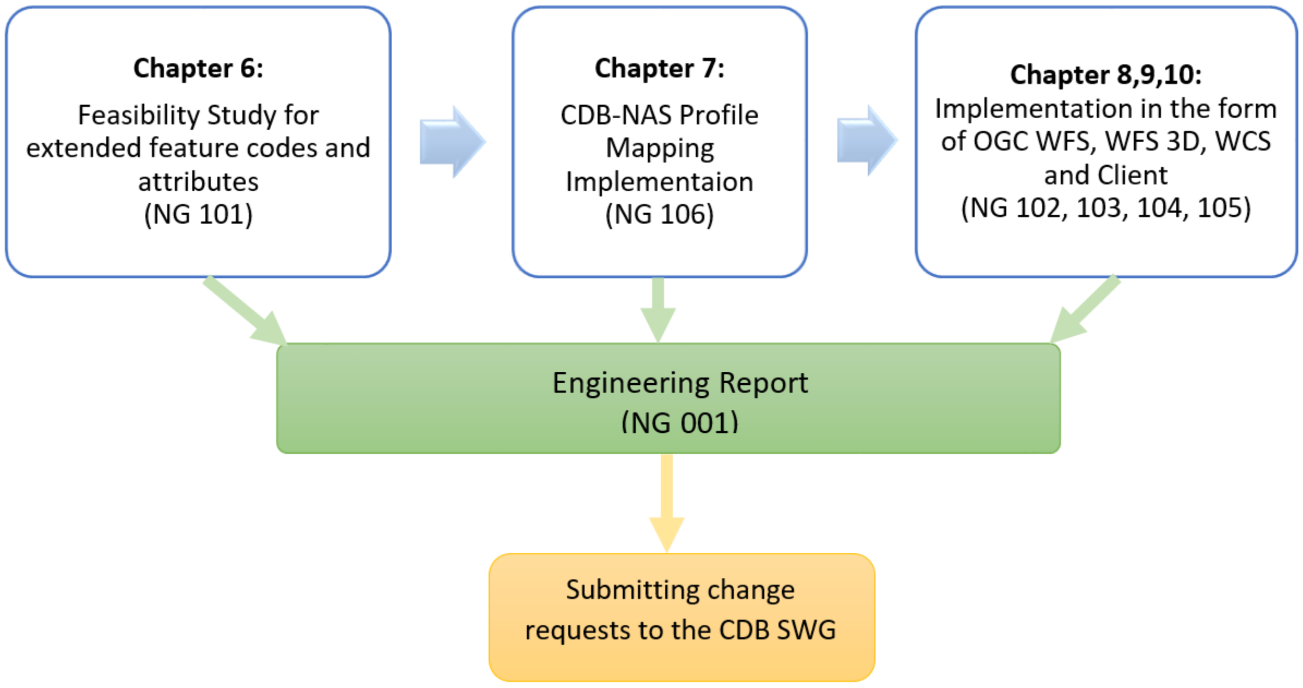

Chapter 6:
Feasibility Study for extended feature codes and attributes (NG 101)
Chapter 7:
CDB-NAS Profile Mapping Implementaion (NG 106)
Chapter 8,9,10:
Implementation in the form of OGC WFS, WFS 3D, WCS and Client (NG 102, 103, 104, 105)
Engineering Report (NG 001)
Submitting change requests to the CDB SWG

Figure 1. CDB sub-thread ER Workflow Outline

This document contains the following chapters:

1. **Preface:** This section presents information on administrative and legal aspects of this ER.
2. **Summary:** This section presents information on scope, what this ER means for the OGC in general and document contributor contact points.
3. **References:** This section presents information on documents that are referenced in this ER.
4. **Terms:** This section presents information on terms and abbreviations that are used in this ER.
5. **Overview:** In this section and overview of the CDB sub-thread work is presented and then, the outline of CDB ER, New York City (NYC) sample dataset, demonstration scenarios and Technology Integration Experiments is discussed.

6. **Background:** This ER includes an overview of the OGC CDB version 1.0 which defines a conceptual model and file structure for the storage, access, and modification of a CDB data store.
7. **Feasibility Study:** The CDB feature codes and attribution schema are introduced at the first step. Also, the feasibility of extending the CDB feature data dictionary is investigated. Pros and cons and limitations resulting from different approaches are documented in this section.
8. **CDB-NAS profile Mapping:** Next, the NAS-based Military Urban Profile is introduced and a mapping between the CDB and NAS is implemented based on the look-up tables. The decision to use a look-up table for transforming feature codes from the OGC CDB to NAS was made for practical reasons, which was to ensure its compatibility with the implementations of OGC CDB 1.0 described in the feasibility study.
9. **CDB WFS:** This section presents information on the CDB WFS implementations, architecture and the results conducted. The CDB WFS is used to access and visualize a sample CDB vector and 3D model data store using a NAS profile feature code and attributes.
10. **CDB WCS:** This section presents information on the CDB WCS implementations, architecture and the results conducted. The CDB WCS is used to access and visualize the tiled elevation and imagery data of a sample CDB datastore.
11. **CDB Client:** This section presents information on the client implementations, architecture and the results conducted.
12. **Findings:** This section summarizes the findings. It also provides forward-looking recommendations.

5.2. Sample Dataset

The sample data set used in the CDB sub-thread is the NYC dataset created by FlightSafety International Inc. solely for use by the National Geospatial-Intelligence Agency (NGA) in support of OGC's Testbed 13 effort. The CDB is physically arranged on disk into the following top level folders:

- **Metadata:** This is a directory structure that contains a set of XML metadata files and controlled vocabularies that are global to the CDB.
- **GTModel:** This is a directory structure that contains Geotypical Models. Geotypical models are a set of generic models that are defined once in the CDB and are intended to be rendered in multiple places throughout the CDB.
- **Tiles:** This is a directory structure that contains the tiled datasets including vector, elevation, imagery and Geospecific model.

The Coordinate Reference System (CRS) of this datastore is WGS 84 (World Geodetic System, 1984). The New York data set contains two CDB Geocells: N40-W74 and N40-W75. The demonstration data store has thirteen tiled data layers: Elevation, MinMaxElevation, MaxCulture, Imagery, GeoSpecificFeatures, GeoTypicalFeatures, VectorMaterial, RoadNetwork, RailRoadNetwork, GeoSpecificModelGeometry, GeoSpecificModelTexture, GeoSpecificModelDescriptorbuildings and Hydro. The CDB sub-thread experiments make use of the Elevation, Imagery, Geotypical, and Geospecific model data layers. Each data layer has its own LoDs (level of Details). A CDB-compliant datastore currently uses ShapeFiles for the representation and attribution of vector feature datasets. Aligned with the capabilities of the ShapeFile format, vector features consist of points, lines and polygon features. The raster data layers such as Elevation and Imagery are in GeoTIFF format. The format of the 3D models, the buildings and trees with geometry & textures, is OpenFlight. OpenFlight is a format widely supported in the modeling and simulation community for dynamic and static 3D models. The texture of those models can be described as .rgb format.

In the NYC CDB data version.xml file, it has been denoted that the dataset is compatible with the CDB 3.2 specification. On the other hand, in the Specification_Version.xml file of the NYC CDB datastore, it was mentioned that the CDB Database Version is 3.0 and released in September 2008. After evaluating most of the datasets inside the CDB it was concluded that this sample data store in Testbed 13 was generated based on CDB 3.0 specifications.

Regardless, there are some enhancements in the OGC CDB 1.0 compared CDB 3.2 specification such as updating the terms and definition to be consistent with the ISO 19xxx (Geomatics) series of standards, specifically ISO 19111 Spatial referencing by Coordinates and ISO 19017 Spatial Schema which is published as the OGC CDB 1.0 Standard, Volume 3: OGC CDB Terms and Definitions [https://portal.opengeospatial.org/files/?artifact_id=72714]. Also, the metadata files and folder updated (*e.g.*, corrections to the Feature Data Dictionary, the addition of a Priority field to the Feature Data Dictionary, additional Base Materials for building interiors, changes to the CDB Attributes, the addition of new airliners and countries codes). Handling of Topological Networks, Model LoDs and significant size and some of the datasets had minor changes as well. The list of these updates can be found in the OGC CDB Wiki page [6]. From the CDB 3.0 to the CDB 3.2 specification, updates are more significant and fully described in the OGC CDB Best Practices documents ([7] and [8]).

NOTE

The sample NYC CDB datastore version is based on an older version (CDB 3.0) and is different from OGC CDB 1.0. The updates and additions are in the form of new datasets, new feature or model attributes, and new enumerations. They are listed below:

- Addition of a CDB Attribute Extension Mechanism
- Changes to FDD entries and fields
- Changes to the CDB Attributes
- Metadata folder and the XML files changed
- Some datasets deprecated and some datasets added
- Model Levels-of-Detail is changed and simplified
- Imagery and Elevation datasets components changed
- Handling of Topological Networks is different
- Additional Base Materials for use with building interiors

5.3. Demonstration Scenario

The underlying use case for the work documented in this report is that:

- Organization **A** as a CDB provider:
  - Wants to share the CDB dataset with other users without them having to download and manually import the data;

    - Should be able to set its CDB as a data source for either a WCS or WFS server application.
- Organization **B** as a CDB consumer:
    - Wants to be able to access a CDB dataset using standardized OGC Web Services protocols;
    - Is able to integrate the CDB data into existing local data analysis workflows using WCS and WFS without having to install or set up CDB-specific components;
    - Wants to access a CDB dataset via a WFS interface with spatial querying capabilities;
    - Wants a client that is capable of viewing CDB datasets either locally or via a WCS/WFS interface;
    - Wants the ability to convert a CDB dataset from CDB feature codes to NAS for a specific purpose and vice versa.

5.4. Technology Integration Experiments

This section discusses the CDB-related Technology Integrated Experiments (TIEs) conducted in Testbed 13.

5.4.1. TIE Components

The following table provides an overview of the components involved in the CDB TIEs.

Table 2. Involved TIE Components

Abbreviation	**Component Name and Description**	**Deliverable**
CDB WCS	Web Coverage Service to access elevation and imagery data	NG103
CDB WMS	Web Map Service to access imagery data	NG103
CDB WFS	Web Feature Service to access vector data layers	NG104
CDB 3DWFS	3D Web Feature Service to access 3D models	NG102
CDB Client	Conversion Client to access both a local CDB and CDB web services	NG105

The above components are available on the University of Calgary component page [https://gsw-app.gswlab.ca/] and the LUCIAD component page [http://demo.luciad.com:8080/OgcCdbServices/#/cdb].

5.4.2. TIE Results

The following table lists all the conducted TIEs.

Table 3. TIE Pairings

TIE between	Use Case	Success
CDB-client / CDB-WCS	Show elevation and imagery data provided by the CDB WCS	Yes
CDB-client / CDB-WMS	Show imagery data provided by the CDB WMS	Yes
CDB-client / CDB-WFS	Show road network data provided by CDB WFS	Yes
CDB-client / CDB-3DWFS	Show 3D vegetation data provided by the CDB 3D WFS	Yes
CDB-client / local CDB loading	Show data from a local CDB datastore	Yes
CDB-client / local CDB NAS conversion	Convert local CDB data to the NAS profile	Yes

The TIEs for the CDB client were all successful and the demonstration videos are uploaded on the OGC portal (OGC Testbed 13*/Streaming and 3D Data/CDB) [https://portal.opengeospatial.org/index.php?m=projects&a=view&project_id=512&tab=2&artifact_id=75970].

Chapter 6. OGC CDB 1.0 Background

The OGC CDB is an international standard for structuring, modeling, and storing geospatial information required in high performance modeling and simulation applications. The CDB open standard enables various application scenarios for visualization, planning, education, training, emergency response and healthcare. CDB defines the core conceptual models, use cases, requirements, and specifications for employing geospatial data in 3D M&S [4]. The main features of the OGC CDB Standard are described as:

- Run-time performance;
- Full plug-and-play interoperable geospatial data store;
- Ability to integrate proprietary and open-source data formats;
- Usefulness in 3D and dynamic simulation environments.

Furthermore, compatibility with the OGC standards baseline reduces the complexity of discovering, transforming, and streaming geospatial data in the synthetic environment and makes them more widely acceptable to major geospatial data/software producers.

6.1. CDB Conceptual Data Model

Conceptual modelling is a structural methodology for describing how the components of a CDB data store will be implemented based on the modular specification design pattern. The OGC CDB conceptual model presents the important components of the core standard (Figure 2). This model, along with its requirements, extension, file-based structure, data formats, access, and the discovery of services, can be used as the basis for the OGC CDB standard in a variety of application domains. The conceptual model is comprised of concepts, schema, classes and categories, as well as their relationships, which are used to understand, and/or represent an OGC CDB datastore. The following figure (Figure 2) describes the CDB original conceptual model and consists of all the UML files related to the CDB's structure. The CDB conceptual model could be implemented in Oracle, PostGIS or almost any other database software.

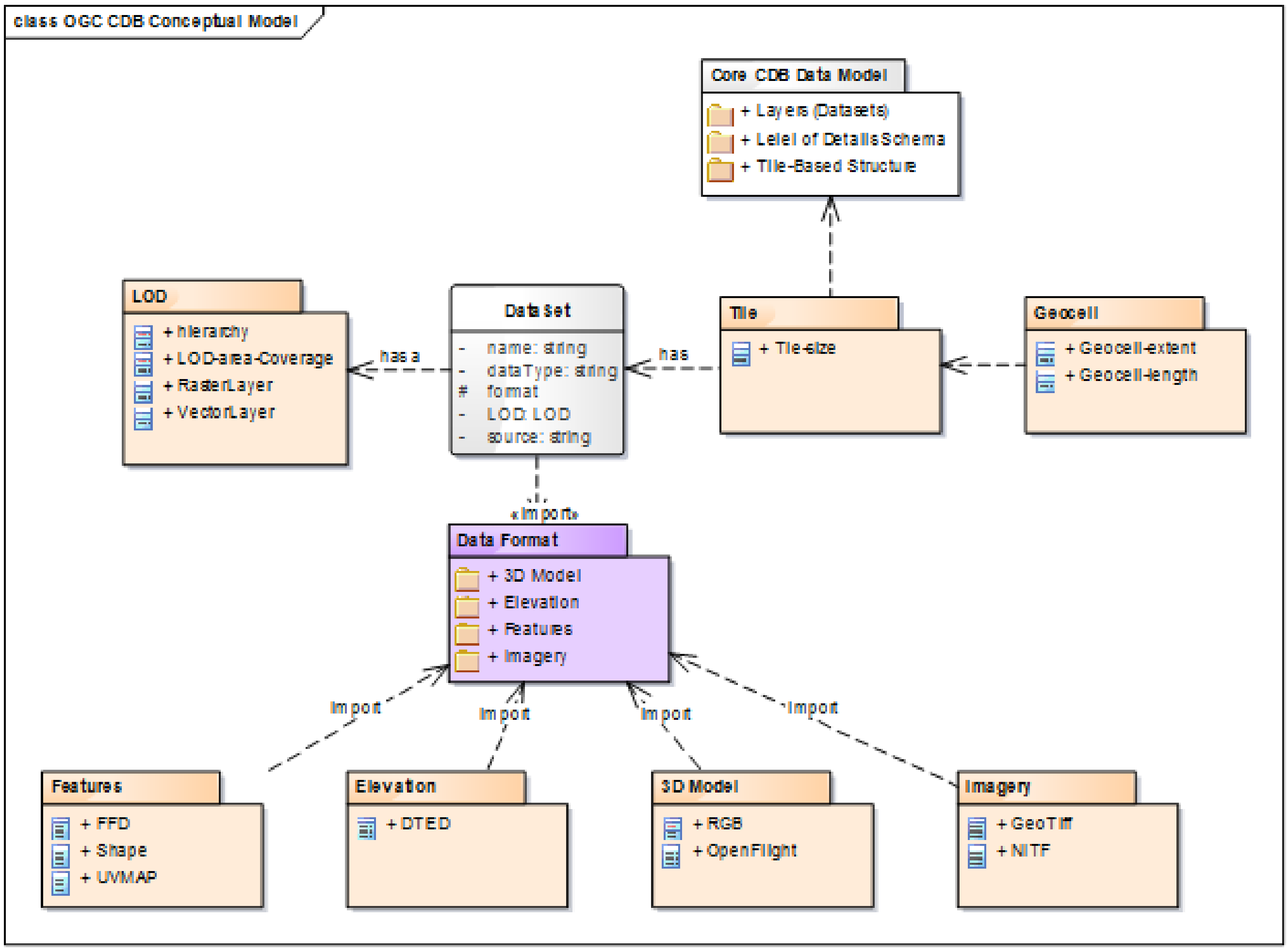


Figure 2. OGC CDB core conceptual model and its related packages

The CDB datastore structure provides efficient access to its contents. The main properties of the CDB datastore UML diagram are listed in the following Table.

Table 4. Description of the OGC CDB conceptual model (Please refer to Figure 2)

Name	Definition	Data Type & Value	Multiplicity & Use
Tile	Geographically divides the world into geodetic tiles (bound by latitudes and longitudes), each containing at least a dataset.	Dataset type.	One or more (mandatory).
LoD Hierarchy	Each dataset layer has a hierarchy of data layers describing different levels of details.	Hierarchy of raster, vector and 3D models.	One or more (mandatory).
Dataset	It defines the basic storage unit used in a CDB datastore.	Layers of data.	One or more (mandatory).

Name	Definition	Data Type & Value	Multiplicity & Use
3D Models	It includes 3D representations of static or moving features such as buildings, pylons and posts, aircraft and other moving platforms. 3D models have various model components.	Model data formats supported in the CDB standard (e.g. OpenFlight).	Zero or more (optional).
Imagery	There are various imageries in a CDB datastore such as representations of geo-referenced terrain, elevation, and texture.	Image data formats supported in the CDB standard (e.g. GeoTIFF, JPEG, etc.).	Zero or more (optional)
Vector Features	These include all the vector feature datasets in a CDB which are defined by the feature codes.	Vectors data formats supported by the CDB (e.g. Shapefile, etc.).	Zero or more (optional).
Elevation	It is depicted by a grid of elevation data elements at regular geographic intervals, or a Triangulated irregular network.	Grid of terrain altimetry data.	Zero or more (optional).
Metadata	A number of CDB XML files that include the default hierarchies, naming, and values to be used by client devices.	XML association.	Zero or more (optional).

The OGC CDB Standard relies on three important means to organize the synthetic environmental data: a) Tiles which organize the data into zones defined by its location with respect to a WGS84 reference system; b) Layers (or datasets) which organize different types of data in a tile; and c) Levels of Detail (LoD) which organize the data in each layer of each tile by its detail. The UML diagram in Figure 3 describes how the data is categorized into tiles, layers and LoDs. This is the basis for the CDB geospatial data categorization. A CDB database can use existing common file formats for storing data such as: TIFF/GeoTIFF (raster data), JPEG 2000 (imagery data), OpenFlight (3D models), Shape (vector data and radar cross sections), RGB (textures), XML (Metadata) and ZIP (file collection).

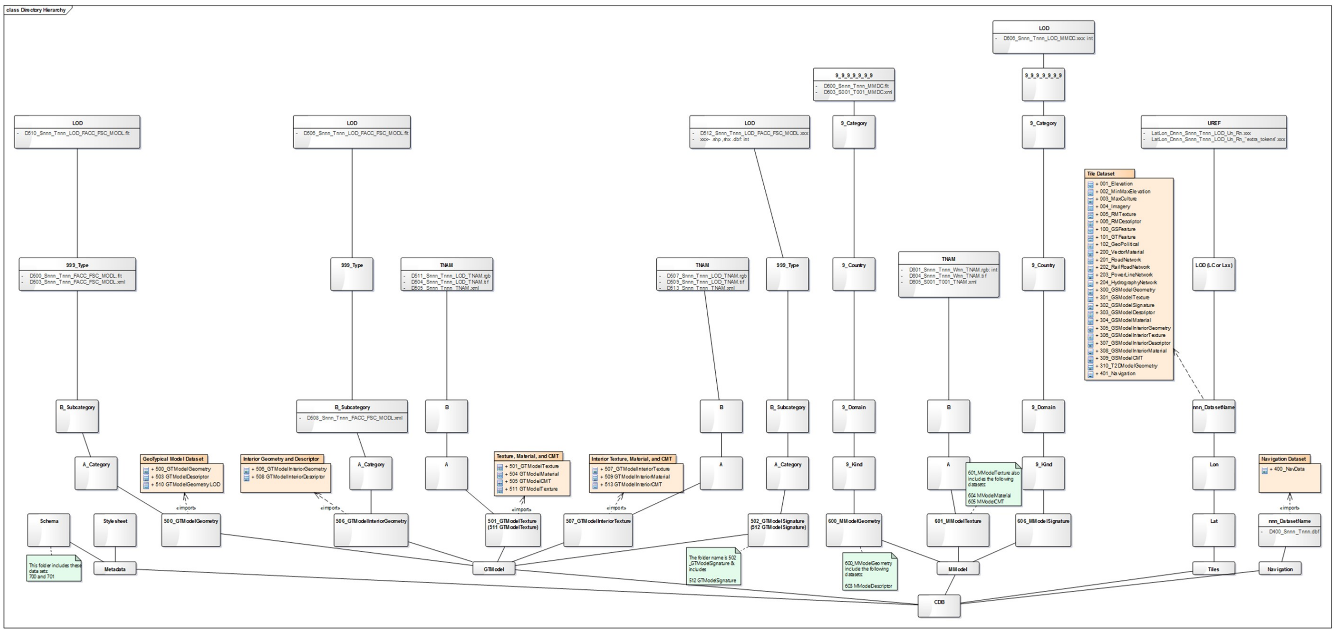

class Directory Hierarchy
Tile Dataset
Geotypical Model Dataset
Interior Geometry and Descriptor
Texture, Material, and CMT
Interior Texture, Material, and CMT
Navigation Dataset
Schema
Stylesheet
Metadata
GTModel
MModel
Tiles
Navigation
CDB

Figure 3. UML diagram of the CDB general data organization

This diagram shows the general data organization for the CDB. The main properties of the CDB general data organization UML diagram are listed in the following table.

Table 5. Description of the OGC CDB data organization (Please refer to Figure 3)

Name	Definition	Data Type	Multiplicity
Raster Dataset	Data elements are organized into an evenly-positioned, regular grid. Raster Datasets always have a fixed number of elements corresponding to their LoD spec.	Raster data formats supported in CDB Core Standard for elevation, imageries, texture and grid data.	Zero or more (optional).
Vector Dataset	The point, line, and area features are organized into several Vector Datasets and LoDs. For each LoD, the maximum number of points allowed per Tile-LoD and the resulting average Feature Density is defined.	Vector data, RMDescriptor, GSFeature, GTFeature, GeoPolitical, VectorMateria, RoadNetwork, RailRoadNetwork, PowerLineNetwork, HydrographyNetwor k.	Zero or more (optional).

Name	Definition	Data Type	Multiplicity
3D Model Dataset	This includes the 3D Geotypical, Geospecific, Moving Model and 2D Model or cultural features such as air platforms, buildings and pylons and posts. 3D models have various model components.	OpenFlight models, GSModelGeometry, GSModelTexture, GSModelSignature, GSModelDescriptor, GSModelMaterial, GSModelInteriorGeometry, GSModelInteriorTexture, GSModelInteriorDescriptor, GSModelInteriorMaterial, GSModelCMT, T2DmodelGeometry.	Zero or more (optional).
Navigation Dataset	Navigation library is composed of a single dataset.	NavData.	Zero or more (optional).

6.2. CDB File Folder Structure

This section describes how the current version of a CDB conformant datastore uses the computer's native file system to store data in files and directories. An important feature of the CDB Standard is that all CDB file names are unique and that the file name alone is sufficient to infer the path of the file. This is an important performance factor for the simulator client to find the path of a specific feature or dataset by its name. The CDB datastore is composed of several datasets that usually reside in their own directory. However, some datasets share a common structure. The top-level directory of the CDB datastore comprises of the following structures:

- *\CDB\:* This is the root directory. It does not need to be named "\CDB\" but can be any valid path name on any disk device or volume under the target file system it is stored on;
- *\CDB\Metadata\:* This directory contains the XML metadata and controlled vocabulary files which are global to the CDB;
- *\CDB\GTModel\:* This is the entry directory that contains the Geotypical Models Datasets;
- *\CDB\MModel\:* This is the entry directory that contains the Moving Models Datasets;
- *\CDB\Tiles\:* This is the entry directory that contains all the tiles within the CDB instance;
- *\CDB\Navigation\:* This is the entry directory that contains the global Navigation datasets.

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

Chapter 7. Feasibility Study of the CDB Extended Feature Codes and Attribute Schema

One of the persistent and important concerns with the OGC CDB 1.0 standard is how to evolve beyond the current use of CDB feature codes and attributes. OGC CDB users and SWG members have expressed an interest in a broader feature data dictionary (FDD) which embodies most (if not all) of the feature codes encountered in the commonly used OGC standards and baselines and at a minimum level, incorporate the much more recent NSG Application Schema (NAS). Therefore, the main objectives of the CDB work package in Testbed 13 are focused on investigating how CDB could support:

- Additional feature types and attributes than those specified in the OGC CDB 1.0 standard;
- Schemas (a specification of the feature types that may occur in a dataset along with their attributes and additional information such as multiplicity and association).

This section explains the current CDB data model, feature coding, and attribution schema. Then, the interoperability issues related to use of the NAS profile are described. Finally, the current solutions for extending CDB feature codes and schema are evaluated and the proposed change requests are presented.

7.1. OGC CDB Feature Codes

The OGC CDB 1.0 standard comes with a set of pre-defined feature types, which are listed in a Feature Data Dictionary (FDD), in the form of an XML file and specified as part of the standard. The list of feature types is defined in a Feature_Data_Dictionary.xml [1] file which is located at */CDB/Metadata/Feature_Data_Dictionary.xml*. The CDB feature dictionary currently does not support a capability to express relationships between features.

NOTE The mostly used formats in CDB to encode feature data (vectors and 3D models) are Shapefiles and OpenFlight files. The CDB feature codes have also been used for indexing Geotypical and Geospecific 3D models.

The OGC CDB 1.0 standard uses a convenient categorization of features (based on FACC code which is now called as "CDB feature code"). The first character in a 5-character CDB feature code (*e.g.*, "CCnnn") represents a category of features, the second represents a subcategory of the current category, and the last three characters represent a specific feature type in the subcategory. To provide an even better classification of features, the CDB standard defines an additional attribute called feature sub-code (FSC). By extending the feature code hierarchy structure in this manner, it is possible to define a broader set of feature types. The sub-code value and its meaning depend on the feature code and varies by feature type. The feature code structure and its data model are presented in Figure 4.

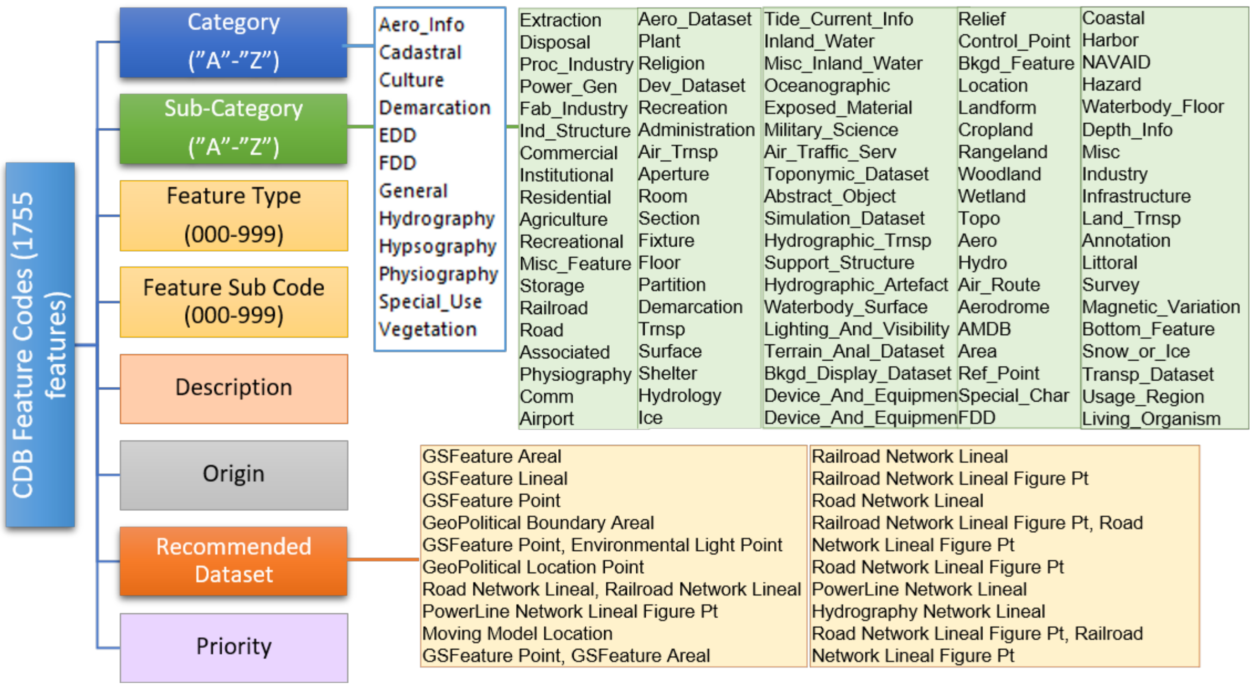


Figure 4. OGC CDB Feature code category and data model

The current CDB FDD is a consolidation of the DIGEST, DGIWG, SEDRIS, and UHRB dictionaries. These standards are commonly used for the attribution of source vector data in a broad range of simulation applications (Figure 5). These Feature Codes were supported by the DGIWG FDD and the ISO/IEC 18025 data dictionaries.

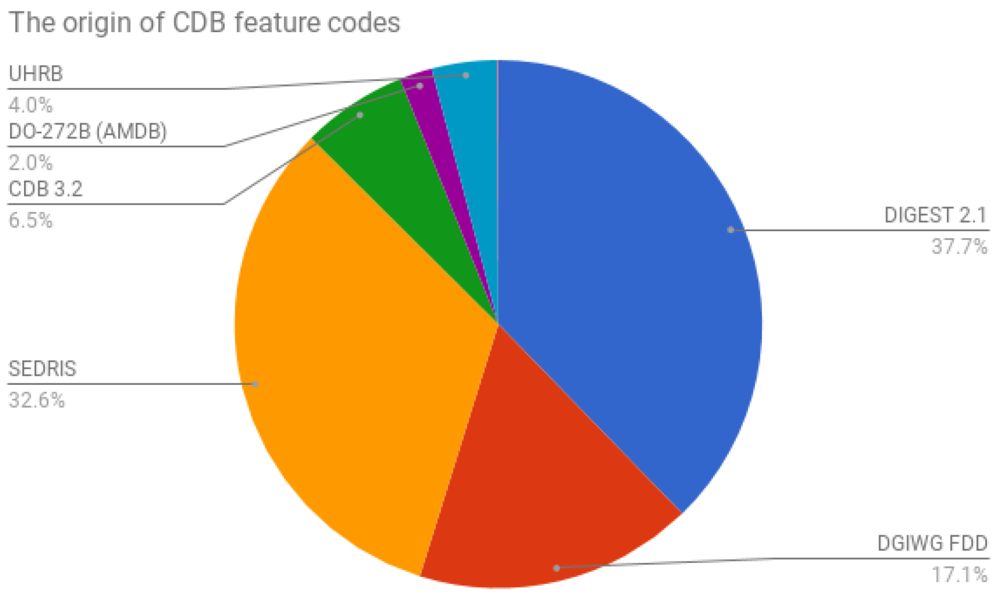


Figure 5. Origin of CDB feature codes

Feature codes such as AL015 are an indexing mechanism for CDB compliant applications to access and navigate to Geotypical models (GTModels) more efficiently. Geotypical datasets (directory, geometry, and description) are likely to be referenced multiple times and are stored under a feature code based hierarchy. Here is an example:

```
\CDB\GTModel\500_GTModelGeometry\A_Culture\L_Misc_Features\015_Building\D500_S001_T001_AL015_050_Church-Gothic.flt
```

The indexing approach greatly simplifies the management of the model library since every model has a pre-established location in the library. The feature codes are also used for naming the files in the CDB folders. In GeoSpecific models, feature codes are used for the file naming in the following data layers: 300_GSModelGeometry and 305_GSModelInteriorGeometry datasets. It looks like:

> *Geocell_DSC_CS1_CS2_LOD_UREF_RREF_FACC_FSC_MODL.ext*

Moreover, feature codes (FACC and FSC) are two attributes for CDB vector features. They are used for allocation of CDB Attributes to Vector Datasets which is called attribution schema.

IMPORTANT

The FDD.xml file allows an application to build the pathname where 3D models associated with features are found. For instance, a feature whose FACC code is *AL015* can be represented by a model that is stored in a folder whose name is *A_Culture\L_Misc_Feature\015_Building*. Without the XML file, the application would have to hardcode the path. Therefore, the CDB feature code is used to index and categorize features within a dataset.

To summarize, the only intent of the FDD file is to provide a programmatic way of building a pathname from a FACC code. The other fields (Concept Definition, Recommended Dataset, and Origin) are for information purposes only. These help applications and users understand the meaning of FACC and FSC codes.

7.1.1. Extending Feature Codes

Although, the list of feature types is fixed and may not be extended in CDB 1.0, broader feature codes can be defined by other OGC standards to be added to the next versions of the standard. For example, since the initial release of the CDB specification and prior to submission to the OGC, 635 new feature codes have been added to the feature data dictionary. The new feature codes use the same coding approach as used in the previous CDB versions, namely a 5-digit FACC supplemented by a 3-digit FSC to be compatible with the other versions of CDB.

IMPORTANT

The FDD.xml has a fixed number of features for the OGC CDB 1.0. In the current structure, the new features can be added by identifying them with a 5-digit FACC code. This identifier code will be used in referencing that feature in the CDB datastore hierarchy.

When a new feature is added to the FDD.xml, it is good practice to denote the recommended dataset property. While the wording is recommended, there is a run time determinism element to this recommendation. If a client wants to find a particular set of features, it can look for all the features in a dataset. Otherwise, it would need to load every data set and search, which defeats a major purpose of CDB.

CDB 1.0 makes extensive use of XML to describe several parts of the standard. XML is used to describe CDB metadata and controlled vocabularies, to store global datasets, to add attributes to data models, to describe base and composite materials and textures, *etc*. Also, it is possible to use a new version of the FDD in CDB using an XML Schema definition based on a service core and an external feature reference instead of a feature data dictionary.

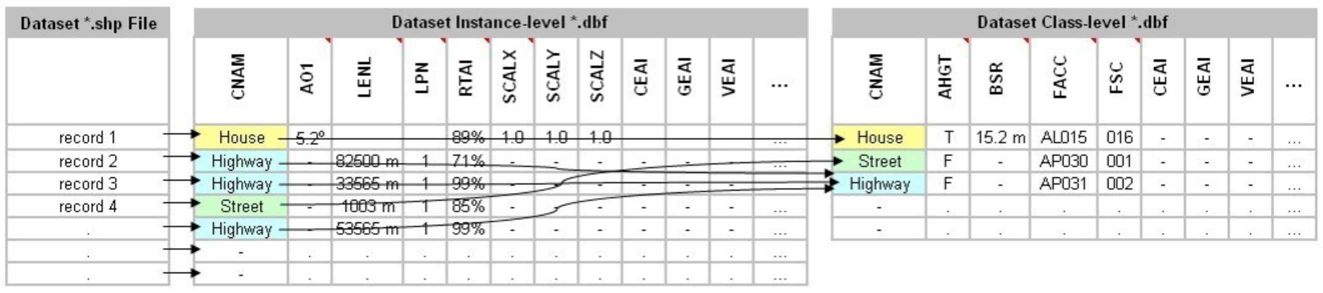
7.2. Attribution Schema and Vector Features

Each vector feature is characterized by a set of attributes that are defined in CDB Attributes [2]. For example, a CDB uses Esri Shapefiles to represent vector data and attributes. As per the Esri Shapefile Technical Description, the set of attributes are stored in dBase III+ files for each instance of the vector feature.

Attribution Schemas are the method to handling these different types of attribution. CDB 1.0 supports three attribute schemas to represent attribution data:

- Instance-level attribute schema
- Class-level attribute schema
- Extended-level attribute schema

Each of the schemas offers different trade-offs in the manner attribution data is accessed and stored. Each of these schemas is largely motivated by storage size considerations and flexibility in the manner attribution data can be assigned to individual features and to groups of features. Attributes are either Mandatory, Optional, not permitted, or not used (Figure 6).



Dataset *.shp File
record 1
record 2
record 3
record 4
.
.
.

Dataset Instance-level *.dbf											
CNAM	A01	LENL	LPN	RTAI	SCALX	SCALY	SCALZ	CEAI	GEAI	VEAI	...
House	5.2°			89%	1.0	1.0	1.0				...
Highway	-	82500 m	1	71%	-	-	-	-	-	-	...
Highway	-	33565 m	1	99%	-	-	-	-	-	-	...
Street	-	1003 m	1	85%	-	-	-	-	-	-	...
Highway	-	53565 m	1	99%	-	-	-	-	-	-	...
-	.	.	.	.	.	.	.	.	.	.	...
-	.	.	.	.	.	.	.	.	.	.	...

Dataset Class-level *.dbf								
CNAM	AHGT	BSR	FACC	FSC	CEAI	GEAI	VEAI	...
House	T	15.2 m	AL015	016	-	-	-	...
Street	F	-	AP030	001	-	-	-	...
Highway	F	-	AP031	002	-	-	-	...
-	.	.	.	.	.	.	.	...
-	.	.	.	.	.	.	.	...

Figure 6. An example of Instance-level and Class-level attribution schema in vector Shapefiles [9]

As the following figure (Figure 7) shows, feature codes (FACC and FSC) are two mandatory attributes for CDB vector features.

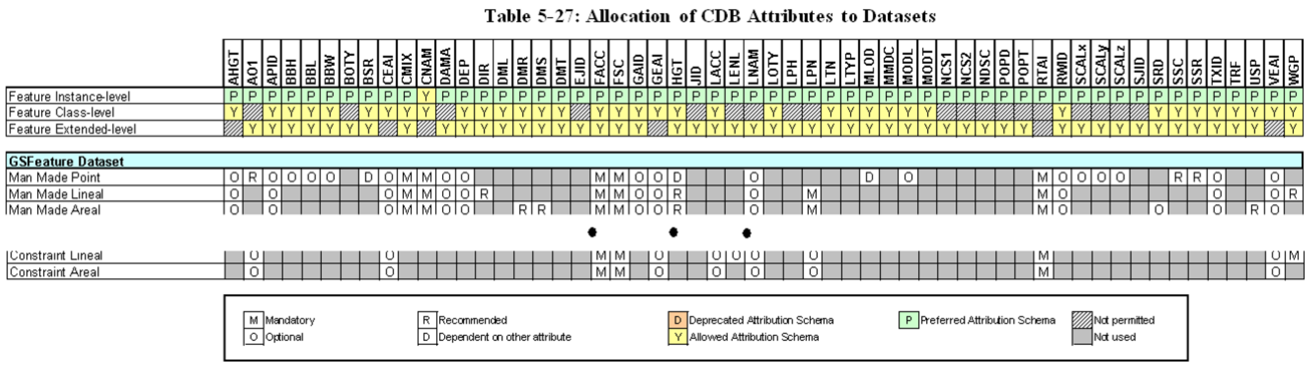


Table 5-27: Allocation of CDB Attributes to Datasets

	AHGT	A01	APID	BBH	BBL	BBW	BOTY	BSR	CEAI	CMIX	CNAM	DAMA	DEP	DIR	DML	DMR	DMS	DMT	EJID	FACC	FSC	GAID	GEAI	HGT	JID	LACC	LENL	LNAM
Feature Instance-level	P	P	P	P	P	P	P	P	P	P	Y	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
Feature Class-level	Y	▨	Y	Y	Y	Y	▨	Y	Y	Y	Y	▨	Y	Y	Y	Y	Y	Y	▨	Y	Y	Y	Y	Y	▨	Y	▨	▨
Feature Extended-level	▨	Y	Y	Y	Y	Y	Y	Y	▨	Y	▨	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	▨	Y	Y	Y	Y	Y
GSFeature Dataset																												
Man Made Point	O	R	O	O	O	O	░	D	O	M	M	O	O	░	░	░	░	░	░	M	M	O	O	D	░	░	░	O
Man Made Lineal	O	░	O	░	░	░	░	░	O	M	M	O	O	R	░	░	░	░	░	M	M	O	O	R	░	░	░	O
Man Made Areal	O	░	O	░	░	░	░	░	O	M	M	O	O	░	░	R	R	░	░	M	M	O	O	R	░	░	░	O
● ● ●																												
Constraint Lineal	░	O	░	░	░	░	░	░	O	░	░	░	░	░	░	░	░	░	░	M	M	░	O	░	░	O	O	O
Constraint Areal	░	O	░	░	░	░	░	░	O	░	░	░	░	░	░	░	░	░	░	M	M	░	O	░	░	O	░	O

	LOTY	LPH	LPN	LTN	LTYP	MLOD	MMDC	MODL	MODT	NCS1	NCS2	NDSC	POPD	POPT	RTAI	RWID	SCALx	SCALy	SCALz	SJID	SRD	SSC	SSR	TXID	TRF	USP	VEAI	WGP
Feature Instance-level	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
Feature Class-level	Y	▨	▨	Y	Y	Y	Y	Y	Y	▨	▨	▨	▨	▨	▨	Y	▨	▨	▨	▨	Y	Y	Y	Y	Y	Y	Y	Y
Feature Extended-level	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	▨	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	▨	Y
GSFeature Dataset																												
Man Made Point	░	░	░	░	░	D	░	O	░	░	░	░	░	░	M	O	O	O	O	░	░	R	R	O	░	░	O	░
Man Made Lineal	░	░	M	░	░	░	░	░	░	░	░	░	░	░	M	O	░	░	░	░	░	░	░	O	░	░	O	R
Man Made Areal	░	░	M	░	░	░	░	░	░	░	░	░	░	░	M	O	░	░	░	░	O	░	░	O	░	R	O	░
● ● ●																												
Constraint Lineal	░	░	O	░	░	░	░	░	░	░	░	░	░	░	M	░	░	░	░	░	░	░	░	░	░	░	O	M
Constraint Areal	░	░	O	░	░	░	░	░	░	░	░	░	░	░	M	░	░	░	░	░	░	░	░	░	░	░	O	░

M	Mandatory	R	Recommended	D	Deprecated Attribution Schema	P	Preferred Attribution Schema	▨	Not permitted
O	Optional	D	Dependent on other attribute	Y	Allowed Attribution Schema			░	Not used

Figure 7. Allocation of CDB attributes to datasets [9]

NOTE

As the above figure shows, the intention and preference in CDB 1.0 are to promote storing all CDB attributes with the instance-level and avoid writing the class-level attributes. Although class attributes are common to several instances of a feature, its optimization does not worth the cost of another read file I/O and it adds complexity to the code. However, for compatibility, class-level attributes are still supported.

CDB consumers have two choices when dealing with class-level attributes:

1. Start by reading the Shapefile of the instance and if a record has a value for the CNAM (class Name) attribute, then read the Shapefile of the class; or
2. Systematically reading both instance- and class level .dbf and dealing with the absence of the class file when none is present.

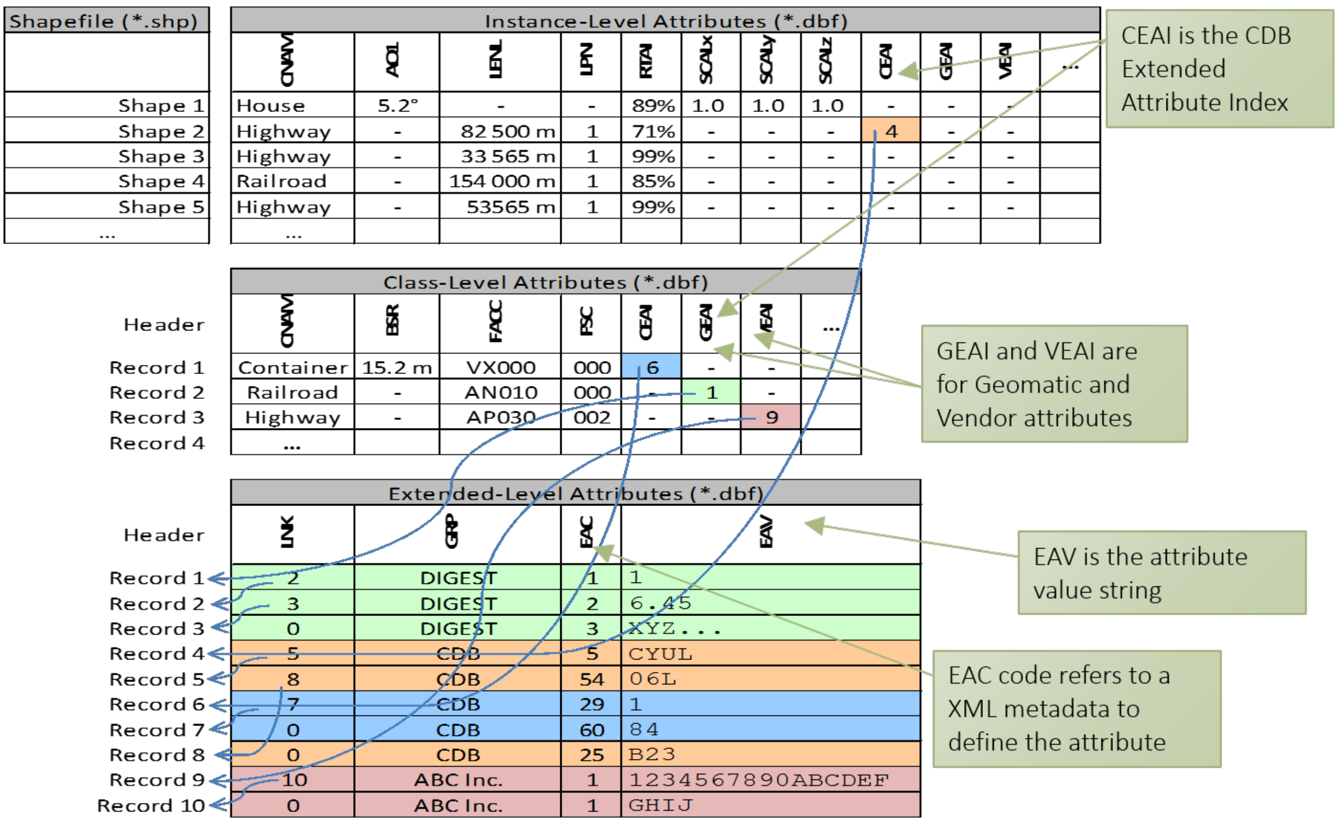
The same reasoning applies to reading the Shapefile containing extended attributes. The user may decide to wait for the presence of values in the CEAI (CDB Extended Attribute Index), GEAI (Geomatics Extended Attribute Index), or VEAI (Vendor Extended Attribute Index) attributes before reading the Shapefile of the extended attributes; or, the user may attempt reading it before parsing the instance or class files in case there are extended attributes referenced by a particular feature. The CDB attribution schema limits the applicability of each of the CDB attributes to specific vector datasets, value ranges and units. This approach helps to reduce the size of the database file of the instance and class-level attribution. This CDB attribute data model is used for the representation of many features using the modeler in real-time simulation.

7.2.1. Extending Vector Feature Attributes

In addition to instance and class level attributes, the standard also promotes using the "Extended-level" attributes which support "Geomatics Attributes" and "Vendor Attributes". The attributes in CDB are categorized into three types:

- *CDB Attributes*: CDB attributes are attributes whose semantics, data type, length, format, range, usage, units, compatibility, and schema are entirely governed by the CDB standard. Most of these attributes are unique to the CDB standard, *i.e.*, these attributes are not found in source data that conforms to various (US) governmental standards and specifications. As a result, this attribution data must be derived via CDB tool automation or provided directly by the user.
- *Geomatics Attributes*: Geomatics attributes are attributes whose semantics, data type, length, format, range, usage, and units, are governed by various governmental/industrial specifications and standards. Such attributes are generally found in source data that conforms to such standards and specifications. While the CDB standard does not define and govern the usage of these attributes, it nonetheless accommodates their storage within the repository structure of the CDB.
- *Vendor Attributes*: Vendor attributes are attributes whose semantics, data type, length, format, range, usage, and units are governed by one or more vendors. In general, such attributes cannot be used by other vendors since they are often proprietary. Such attributes exclude the above two types of attributes and are generally unique to each vendor. While the CDB standard does not define and govern the usage of these attributes, it nonetheless accommodates their storage within a CDB.

There are three XML files that define CDB, Geomatics, and Vendor Attributes. Only one is provided with the CDB distribution package. This one file is "/CDB/Metadata/CDB_Attributes.xml". The other two are necessary only if Geomatics or Vendor attributes are used to create Extended Attributes (see 5.7.1.2.7.3 of Volume 1). Geomatics_Attributes.xml and Vendor_Attributes.xml define the attributes that are referenced in the EAC column of the Extended Attributes tables (see Figure 8). The intent behind those XML Attribute files is to provide the data necessary to interpret EAC (Environment Attribute Code) and EAV (Environment Attribute Value). For instance, if EAC = 1 (numeric) and EAV = “123.456” (text), the XML Attribute files will specify what is the name and type of attribute 1 so that the character string “123.456” can be interpreted as a floating point number.


CEAI is the CDB Extended Attribute Index
GEAI and VEAI are for Geomatic and Vendor attributes
EAV is the attribute value string
EAC code refers to a XML metadata to define the attribute

Shapefile (*.shp)	CNAM	AO1	LENL	LPN	RTAI	SCALx	SCALy	SCALz	CEAI	GEAI	VEAI	...
Shape 1	House	5.2°	-	-	89%	1.0	1.0	1.0	-	-	-	
Shape 2	Highway	-	82 500 m	1	71%	-	-	-	4	-	-	
Shape 3	Highway	-	33 565 m	1	99%	-	-	-	-	-	-	
Shape 4	Railroad	-	154 000 m	1	85%	-	-	-	-	-	-	
Shape 5	Highway	-	53565 m	1	99%	-	-	-	-	-	-	
...	...											

Class-Level Attributes (*.dbf)

Header	CNAM	BSR	FACC	FSC	CEAI	GEAI	VEAI	...
Record 1	Container	15.2 m	VX000	000	6	-	-	
Record 2	Railroad	-	AN010	000	-	1	-	
Record 3	Highway	-	AP030	002	-	-	9	
Record 4	...							

Extended-Level Attributes (*.dbf)

Header	LNK	GRP	EAC	EAV
Record 1	2	DIGEST	1	1
Record 2	3	DIGEST	2	6.45
Record 3	0	DIGEST	3	XYZ...
Record 4	5	CDB	5	CYUL
Record 5	8	CDB	54	06L
Record 6	7	CDB	29	1
Record 7	0	CDB	60	84
Record 8	0	CDB	25	B23
Record 9	10	ABC Inc.	1	1234567890ABCDEF
Record 10	0	ABC Inc.	1	GHIJ

Figure 8. An example of extended-level attribution schema in vector Shapefiles [9]

The schema for all of the three XML files which define CDB, Geomatics, and Vendor Attributes can be found in "\CDB\Metadata\Schema\Vector_Attributes.xsd". The file contents are presented as follows:

```
<Vector_Attributes>
  <Attributes>
    <Attribute>...</Attributes>
      ...
    <Attribute>...</Attributes>
  </Attributes>
  <Units>
    <Unit>...</Unit>
    ...
    <Unit>...</Unit>
  </Units>
  <Scalers>
    <Scaler>...</Scaler>
    ...
    <Scaler>...</Scaler>
  </Scalers>
  </Version>
```

The following example illustrates how to define an attribute:

```xml
<Vector_Attributes>
  <Attributes>
    <Attribute code="2" symbol="AO1">
        <Name>Angle of Orientation</Name>
        <Description>Angle of Orientation with greater than 1 degree resolution.
          The angular distance measured from true north (0 degree) clockwise to the
          major (Y) axis of the feature. If the feature is square, the axis 0
          through 89.999 degree shall be recorded. If the feature is circular, 360.000
          deg shall be recorded.
        </Description>
        <Level>
          <Instance>Preferred</Instance>
          <Extended>Supported</Extended>
        </Level>
        <Value>
          <Type>Numeric</Type>
          <Format>Floating-point</Format>
          <Precision>3.3</Precision>
          <Range interval="Right-Open">
              <Min>0</Min>
              <Max>360</Max>
          </Range>
          <Unit>2</Unit>
        </Value>
      </Attribute>
    </Attributes>
    <Units>
      <Unit code="2" symbol="deg">
      <Name>degree</Name>
      <Description>To measure an angle</Description>
    </Unit>
  </Units>
  <Scalers>
      <Scaler code="2" symbol="k">
      <Name>kilo</Name>
      <Description>A multiplier: thousand</Description>
      <Multiplier>1000</Multiplier>
  </Scaler>
</Scalers>
</Vector_Attributes>
```

7.3. Supporting Application Schema

An application schema includes feature type and model definitions (code, attributes, units, etc.) along with any constraints important for the integrity of the data. Currently, CDB only defines a simple attribute schema for vector features which at the implementation level can be stored in the Shapefile .dbf files. When considering adding support for application schemas in CDB, several aspects need consideration. This is extensively studied in Testbed 13 NAS Profiling ER [3]. These concerns are related to the following topics:

1. *Schema Encoding*: The CDB 1.0 encoding of the application schema can be implemented using the Shapefiles. The current CDB 1.0 attribute schema is also implemented using Shapefiles. The NAS specifies encodings for GML and Esri Geodatabases. Since no NAS data was used in CDB in Testbed-13, no recommendations about future encodings were discussed in Testbed-13. In future, other encodings such as GML, Esri Geodatabases, and GeoPackage may be used.
2. *Schema description*: The CDB 1.0 schema description is implemented using three tables (`.dbf` files) to store the attribute schema. The columns which describe the CDB and extended attributes are linked to the corresponding `.xml` to describe them. In Testbed 13, the NAS profile was expressed as a GML application schema to support simple schemas (GML Simple Feature Level 0 profile) for CDB. Although this simplified version has some limitation in using the complex content of the data model, the decision was made for practical reasons: to be compatible with the complexity supported by Shapefiles and, therefore, implementations of OGC CDB 1.0.

If a CDB data store is to be able to support application schemas, the standard needs to clarify if the feature dictionary and attribute schema are still required and what is their role. The CDB feature data dictionary is currently used in CDB for indexing and accessing the features.

7.4. Limitation of the OGC CDB 1.0

CDB is a file-based data store designed to access and visualize data at different levels of abstraction. It has an internal/physical schema for the indexing of folders and file names for random access. This system is useful for fast access which affects performance, but not semantics. Due to the fixed rules for file naming and file path definition, rapid implementation of new features and changing the indexing structures is difficult to define and implement. However, useful routines can be hardcoded or represented in `xml` to deal with the issue of physical representation.

Designing a method for supporting additional feature codes and schema should also maintain backwards compatibility as best as possible. Data does nothing in the absence of an interpreter (such as a database generation tool or a client device). As a result, the notion of compatibility should apply not only to CDB itself, but also to the simulation and modeling software that implement the CDB standard. There are actually two types of compatibility that should be considered:

- *Backward compatibility*: refers to the ability of an interpreter implemented to version n of the standard to accept a CDB compliant to version (n-1) of the standard. Logically, if version (n-1) is also backward compatible with version (n-2), which in turn is backward compatible with version (n-3), then version n is backward compatible with the oldest version that is not backward compatible with its predecessor.
- *Forward compatibility*: refers to the ability of a CDB datastore to accept input intended for a later version of itself and pick out the "known" part of the data. Forward compatibility is harder to achieve than backward compatibility because software needs to cope smoothly with an unknown future data format or requests for unknown future features.

The other important factor with the CDB standard is the performance issue associated with the extended attributes or additional feature data dictionary. Since all the data sources in a CDB data store need to use extended feature attributes, there will be a performance bottleneck in run-time implementations. Therefore, addressing a method for extended feature attributes should address this issue.

NOTE Designing a method to support extended feature codes/attributes and schemas in the OGC CDB 1.0 should address the run-time performance and backward/forward compatibility as much as possible.

7.5. Extending CDB 1.0 Using NAS Profile in Testbed 13

The compatibility of the CDB standard with newer versions of feature codes (such as NAS) makes them widely acceptable to major geospatial data/software vendors and stakeholders. Furthermore, this provides a tool for extending the CDB feature codes/attributes and reduces the complexity of discovering, transforming, and streaming geospatial data from the Internet to the simulation environment.

Since the CDB feature codes (based on FACC codes) are ingrained into the structure and naming scheme of the CDB 1.0, the proposed approach in Testbed 13 is based on keeping the current codes as is. In the Testbed 13 CDB sub-thread, another method was applied to convert CDB feature codes and attributes to the NAS profile for accessing and visualizing the New York City CDB sample data. For this purpose, three mapping tables are considered for three feature data sources: tiled vector, Geospecific and Geotypical model dataset. Using these look-up tables, the client device can find the corresponding feature in the NAS profile for each feature instance in CDB vector, Geospecific or Geotypical model datasets. The details of this approach are described in chapter 8 of this ER.

One possible approach for adding NAS attributes to CDB 1.0 is to insert them as the extended-level attributes into the feature attributes. For example, a new attribute can be added to the extended level attribute column of the `.dbf` files and described by a generated `GeomaticAttribute.xml` file from NAS profile. This method is carefully investigated in [3].

7.6. Proposed Change Requests for the OGC CDB 2.0

A practical short-term solution was applied in Testbed 13 to extend the current OGC CDB 1.0 features and attributes (please see the above section). This ER recommends that additional work is required for the replacing of CDB feature codes and attributes. The following future work and changes are expected to be proposed for future CDB versions.

One important change to CDB 1.0 is replacing the feature codes for CDB and updating them with the current technology for complex data modeling, associations and semantics. For example, one solution is to replace the feature data dictionary with Application Schemas (*e.g.*, NAS [3]) that describe the features and attributes in different domains of application with their associations and relationships. To extend the current simple schema of the CDB using NAS (SF0), the NAS feature types are converted into a simple format which is accepted and encoded for the OGC CDB 1.0. Then, these NAS attributes can be added to the feature as an extended attribute.

The critical challenge here is that the indexing technique encoded in CDB (for example the Geotypical model indexing) is based on the FACC codes. Preferably this constraint should be removed. Tis recommendation is based on the discussions in Testbed 13. Another compatible approach may use a mapping to look-up the deprecated FACC codes for each feature type in the

CDB.

Change Proposal 1:

How to support application schemas in CDB by replacing FACC codes with a general feature code?

If future versions of CDB standard are intended to support the NAS (or any other application schemas), defining a more general feature code is necessary to support the feature types used by the application schema. This feature code should be able to support:

- A more flexible grouping of feature types (1 .. n category levels) instead of two Category and Sub-Category levels. Although it is similar to NAS two-level grouping, this will not be the case for other application schemas.
- Optional `Subcode` for the features as it does not have a corresponding concept in the General Feature Model, which underpins ISO 19109 application schemas like the NAS.
- Changing some of the properties of the FDD (*e.g.*, description, recommended DB), as the *description* field is equal to the *definition* in NAS and the *recommended DB* is only useful for high-performance simulations.
- Performance testing to see if different FDD can be supported in real time.

In Testbed 13, encoding of the application schema can be implemented using the `Shapefiles .dbf` files for vector features. Moreover, for the 3D visualization performance, one of the difficulties was reading the 3D OpenFlight models. In the NYC CDB datastore, OpenFlight is used for the representation of 3-dimensional geometric representation of all models. Therefore, the following change request is proposed for CDB 1.0 by participants in all of the S3D performance sub-threads.

Change Proposal 2:

How to use open formats in the CDB for vectors and 3D models?

- Support acceptance of the frequently used format "OpenFlight" as an OGC community standard.
- How to use CityGML in OGC CDB?
- How to use CDB with GeoPackage?

The above changes may require performing OGC interoperability experiments to better understand the implications of these decisions better. Making these enhancements will allow for the use and implementation of a CDB structured datastore and the extension of its scope to include more 3D simulation and modeling application.

Chapter 8. CDB to NAS Mapping for Feature Codes and Attributes

The proposed method used in Testbed 13 for interoperability between CDB and NAS was to find a mapping and look-up table for the features to access NAS-based data in the CDB data store or save CDB-encoded attributes to a CDB NAS profile format. This section describes some examples of the mapping between CDB features (vector and 3D models) and the NAS profile. For this purpose, three types of features in a CDB data store need to be mapped to the NAS profile: Geotypical 3D models, Geospecific 3D models, and vector features (points, lines and polygons). In Testbed 13, a NYC CDB data store has been used which has 3D models (building and trees) in the Openflight format and vectors (road, railway network, rivers, lakes, hydrology, and etc.) in the Shapefile format. In this section only one example of "Tree" 3D model feature is described. For more details on the mapping lookup tables please see "CDB to NAS Mapping for Feature Codes and Attributes", document 17-082 [5].

TIP How to obtain the NAS profile formatted feature code and attributes from a feature instance inside a CDB?

To answer the above question, the user first needs to find out if that feature is a Geospecific, Geotypical 3D model or Tiled vector feature. After that, the considered feature type can be found in the corresponding detailed mapping table. For example, if a "maple tree feature" instance is considered, it is a Geotypical feature that can be a point or area feature. Then the "maple tree" feature code (FACC code) is extracted from its filename or from the CDB 1.0 `FDD.xml` file. Based on the height of the feature, there is a Geotypical model associated with it which encoded the feature code in its naming convention. The maple tree feature instance is mapped to its equivalent NAS profile formatted feature using that feature code. The OGC Testbed 13 NAS profile workbook keeps the deprecated FACC codes that are used as feature codes in a CDB data store.

8.1. NAS Urban Military Profile

The National System for Geospatial Intelligence (NSG) Application Schema (NAS) [https://nsgreg.nga.mil/nas/] specifies a platform independent model for geospatial data and their geospatial semantics specified by the NSG Entity Catalog (NEC) and NSG Feature Data Dictionary (NFDD). The NAS conforms to ISO 19109:2005 Rules for Application Schema, as well as, conceptual schemas specified by other ISO 19100-series standards. NAS includes entity modeling for modeling features, events, names and coverages (*e.g.*, grid, raster, and TIN).

As the NAS specifies an NSG-wide model for geospatial data that supports a wide variety of domains and applications, it is advantageous to define subsets of the NAS that meet specific requirements.

In order to extend CDB using NAS in Testbed 13, an "Urban Military Profile" of the NAS was used as a starting point. This profile defines the vector content requirements for urban-centric profiles. This profile is intended to be used to define the data model requirements for an Urban Military CDB. The S-UTDS subset of the TDS v6.0 Entity Catalog [https://nsgreg.nga.mil/doc/view?i=82116] (topographic-profile of an older version of the NAS) has been used as the starting point for the "Urban" Military Profile of the NAS for the purposes of this testbed [3]. NAS has been used as an example of the recent feature data models which includes geospatial data semantics; supports net-centric geospatial services; and is capable of achieving geospatial data interoperability [3].

8.2. Mapping of "3D Geotypical Model" Feature Codes

The 3D model of trees is located in the NYC CDB datastore as a Geotypical model. The Geotypical model is composed of a set of datasets representing its geometry (ModelGeometry and ModelDescriptor) and its textures (ModelTexture, ModelMaterial, and ModelCMT). Similarly, the interior of a model is divided into geometry (ModelInteriorGeometry and ModelInteriorDescriptor) and textures (ModelInteriorTexture, ModelInteriorMaterial, and ModelInteriorCMT) datasets. Here is the path of the Geotypical models of the NYS dataset which are stored in: *CDB_NewYork\GTModel\500_GTModelGeometry\E_Vegetation\C_Woodland\030_Trees*

In Geotypical model, the indexing approach greatly simplifies the management of the model library since every model has a pre-established location in the library. The following table describes the CDB-NAS feature code mapping for the 3D model of maple trees.

Table 6. Mapping between CDB and NAS feature codes for 3D Geotypical model of maple trees

NAS Profile (deprecated Feature ID)	CDB Feature Code	CDB Feature Sub-Code	CDB Path and File Name (CDB_NewYork\GTModel\500_GTModelGeometry\E_Vegetation\C_Woodland\030_Trees)
EC005	EC030	000	D500_S001_T001_EC030_000_central_park_maple.flt
EC005	EC030	000	D500_S001_T002_EC030_000_central_park_maple.flt
EC005	EC030	000	D500_S001_T003_EC030_000_central_park_maple.flt
EC005	EC030	000	D500_S001_T001_EC030_000_maple_medium_40ft.flt
EC005	EC030	000	D500_S001_T001_EC030_000_maple_medium_45ft.flt
EC005	EC030	000	D500_S001_T001_EC030_000_maple_medium_50ft.flt
EC005	EC030	000	D500_S001_T001_EC030_000_maple_medium_60ft.flt
EC005	EC030	000	D500_S001_T001_EC030_000_maple_medium_70ft.flt
EC005	EC030	000	D500_S001_T001_EC030_000_maple_medium_80ft.flt

As is presented in table 5, maple trees have different 3D models for different height values. With a closer look into the CDB feature data dictionary file (FDD.xml), the following feature codes are associated with maple trees.

Table 7. OGC CDB 1.0. feature codes and related information for maple tree features in the FDD.xml file

CDB Feature Code	CDB Feature Sub-Codes	Labels	Concept Definition	Recommended dataset	Origin
EC030	000	Vegetation-Woodland-Tree-Trees_Generic	A tract of trees whose canopy is not closed (allows sunlight to reach the ground) and often includes undergrowth. (See also EA040, EB020 and EC015)	GSFeature Areal	DIGEST 2.1
EC030	033	Vegetation-Woodland-Tree-Maple	A tract covered by a non-empty set of Maple trees; a treed tract.	GSFeature Areal	DIGEST 2.1
EC005	000	Vegetation-Woodland-Tree-Tree_Generic	An individual woody perennial plant, typically having a single stem or trunk growing to a considerable height and bearing lateral branches at some distance from the ground.	GSFeature Point	DIGEST 2.1
EC005	033	Vegetation-Woodland-Tree-Maple	A individual tree of kind Maple.	GSFeature Point	CDB 3.2

The above table shows that in the OGC CDB 1.0 which inherited the CDB 3.2 specification, the trees are extended to include EC005 which is a point feature instead of the EC030, which is a polygon feature.

IMPORTANT The Testbed 13 NYC dataset is based on the CDB 3.0 specification and denoted the "maple trees" only with the EC030.

The following table shows the feature codes in NSG application schema. There are two fields for defining the features: *Definition* and *Description*. The field *Definition* from the NAS profile is related to the field *Concept Definition* in CDB. Various types of trees can be defined in the Tree attributes.

Table 8. Feature definition in NSG application schema

AlphaCode	Name	Definition	Description
Tree	Tree	An individual woody perennial plant, typically having a single stem or trunk growing to a considerable height and bearing lateral branches at some distance from the ground.	May be distinguished by its relative isolation from other features, thus serving as a landmark.

8.3. Mapping of "3D Geotypical Model" Attributes

Another critical discussion for the CDB to NAS mapping is the attribute schema. When the equivalent features in the CDB and NAS are found, the attributes of these features can be very different. NAS has more information about the attributes, association, geometry and their relationships.

For example, the following table shows the properties of the *Tree feature* in the NAS profile. This is generated based on the Testbed 13 NAS profile.

Table 9. Flattened NAS Profile feature entity code, attributes, enumeration of tree feature

Property Name	**Property Code**	**Type**	**Enumeration**	**Association Information & Unit**
Place	Tree. place	*geospatial	1..*	Point Position Information (ZI007)
Foliage Type	EC005_TRE	attribute	0..1	
Height Above Surface Level	EC005_HGT	attribute	0..1	Unit: metre
Highest Elevation	EC005_ZVH	attribute	0..1	Unit: metre
Navigation Landmark	EC005_LMC	attribute	0..1	Domain members: False, True
Subclass-of	EC005_subclassOf	attribute		Feature Entity {Abstract} (ZI028)

For more descriptions about NAS attributes, please see [5].

From the CDB attribution schema (`Attributes.xml`), it can be seen what are the definitions for attributes and which attributes are deprecated (D), supported (S), not supported (NS) or preferred (P).

Table 10. Attribute schema for "maple tree" from “CDB_Attributes.xml” file (S: Supported, P: Preferred, N S: Not-Supported)

Attribute			**Level**			**Value Type**
Full Name	**Symbol**	**Code**	**Instance**	**Class**	**Extended**	**Range and Unit**
Absolute Height Flag	AHGT	2		S		Boolean
Angle of Orientation	AO1	3	P	N S	S	Numeric Floating-Point, Range [0, 360], Unit(deg)
Bounding Box Height	BBH	6	N S	P	S	Numeric Floating-Point, Range [0, 100000], Unit(m)
Bounding Box Width	BBW	7	N S	P	S	Numeric Floating-Point, Range [0, 100000], Unit(m)

Attribute			Level			Value Type
Bounding Box Length	BBL	8	N S	P	S	Numeric Floating-Point, Range [0, 100000], Unit(m)
Bounding Sphere Radius	BSR	10		P		Numeric Floating-Point, Range [0, 100000], Unit(m)
Composite Material Index	CMIX	13		S	P	Numeric Integer, Range [0, 999999]
Class Name	CNAM	14	S	S		Text (32 char)
Feature Attribute Classification Code	FACC	23		S		Text (5 char)
FACC Sub Code	FSC	24		S		Numeric Integer, Range[0, 999]
Height Above Surface Level	HGT	27		S		
Model Name	MODL	40		S		Text (32 char)
Model Type	MODT	41		S		Text (1 char): Geotypical or Geospecific
Relative Tactical Importance	RTAI	53	S			Numeric, Integer, Range[0, 100], Unit(%)
Scaling X-Axis	SCALx	55	S		P	Numeric Floating-Point, Range[0, 1000]
Scaling Y-Axis	SCALy	56	S		P	Numeric Floating-Point, Range[0, 1000]
Scaling Z-Axis	SCALz	57	S		P	Numeric Floating-Point, Range[0, 1000]

For more descriptions about CDB attributes, please see [5].

In the above table, these abbreviations used for determining when an attribute is required for instant/class/extended level: S=Supported, P=Preferred, N S=Not Supported.

NOTE As the above tables show, the intention and preference of the OGC CDB 1.0 is to promote storing all CDB attributes with the instance-level and avoid writing the class-level attributes. Although class attributes are common to several instances of a feature, its optimization does not worth the cost of another read file I/O and it adds complexity to the code. However, for compatibility, class-level attributes are still supported.

For a tree’s Geotypical models, the following tables show some examples of the CDB attributes in class level and instant level.

Table 11. An example of OGC CDB 1.0. class level attributes for maple tree features

Class Name (CNAM)	Scale in x (SCALx)	Scale in y (SCALy)	Scale in z (SCALz)	Angle of Orientation (AO1)	Relative Tactical Importance (RTAI)
N40W074L00U000R000~EC030~0000005	1.19218	1.19218	1.19218	129.170	0

Table 12. An example of OGC CDB 1.0. instance level attributes for maple tree features

Class Name(CNAM)	Model	FACC	FSC	BSR	BBH	BBW	BBL	HGT	CMIX	AHGT
N40W074L00U000R000~EC030~0000005	maple_medium_80ft	EC030	0	21.119	24.386	24.386	24.386	24.39	1	F

Chapter 9. CDB WFS

The Web Feature Service (WFS) [http://www.opengeospatial.org/standards/wfs] is an OGC standard defined as a web service interface for the exchange of geospatial vector data. The vector data is organized into "feature types", each having a set of similar vector features. The default data exchange format is a GML [http://www.opengeospatial.org/standards/gml] application schema. In order to interact with a WFS instance (endpoint) and consume data, a WFS client can perform the following operations:

- The *GetCapabilities* operation is a request to a WFS server to obtain a list of operations and feature types offered by that server;
- The *DescribeFeatureType* operation returns the GML application schema for a selected feature type. The returned XML Schema can be used by a WFS client to determine the structure of the data (properties, value types, etc.) and to support data parsing;
- The *GetFeature* operation returns the selected feature type encoded using a GML application schema. Clients can optionally filter the requested feature type by supplying an OGC Filter [http://www.opengeospatial.org/standards/filter], which can contain filters based on property values, geospatial coverage, temporal extent or a combination of them.

Additional operations can be defined by the WFS standard to support more advanced capabilities such as stored query management and transactions.

The CDB WFS in Testbed 13 focuses on delivering vector and 3D data from CDB-related data sources. The following sections discuss in more detail the architecture, design and implementation of the CDB WFS.

9.1. Architecture & Design

This section describes the architecture of the CDB (3D) WFS and the design decisions taken to support the envisioned use case. Figure 9 shows the high-level architecture of the WFS. The CDB standard relies on common data formats for its various data types. In case of vector and 3D data, this respectively includes Shapefile and OpenFlight (geometry) and SGI (texture). The contributed CDB (3D) WFS is able to read and serve this data directly from a CDB datastore without any preprocessing. A WFS client should be able to access and visualize the data and align it with the WFS standard without any loss of information residing in the CDB datastore.

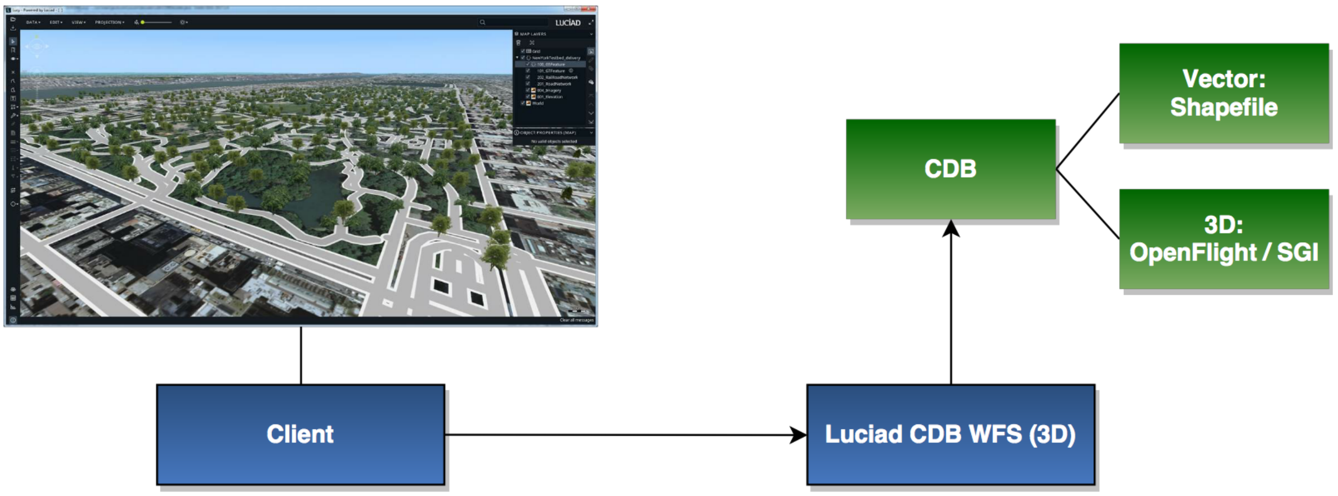


Figure 9. CDB (3D) WFS system architecture

Most CDB datasets are organized in a tile structure, following the CDB tiling scheme described in the OGC CDB Core Standard [https://portal.opengeospatial.org/files/?artifact_id=72712] (Volume 1 - document #15-113r3, Section 3.1 and 3.6). Within a tile, data is organized into separate datasets, each including one or more levels of detail. Within OGC Testbed 13, the focus is on offering access to tiled (3D) vector data via the WFS. Although other CDB data types, such as moving models and navigation data were not considered, this is not expected to introduce additional complexity, given the fact that similar data formats were used to represent the data in CDB.

9.1.1. Organizing Tiled, Multi-Leveled Data in a WFS

By definition, a WFS instance has the ability to organize data into feature types, but this does not include the notion of tiles or multiple levels of detail. By using an OGC Filter, a WFS client can spatially filter the desired data, but this is unrelated to a tiling scheme: in the simplest form, the coordinates of an area of interest are supplied by the WFS client and used by the WFS server to filter the requested feature type data. Examples of more advanced spatial filtering cases include the ability to test based on topological relationships, and to filter out data that is located within a distance of a given feature.

Based on the above, a logical approach to organize and offer CDB tiled data via a WFS is to map CDB datasets onto WFS feature types (e.g. *RoadNetwork*). In this case, each WFS feature type groups all available data of that CDB dataset for a predefined level of detail. Figure 10 shows a schematic overview of this approach.

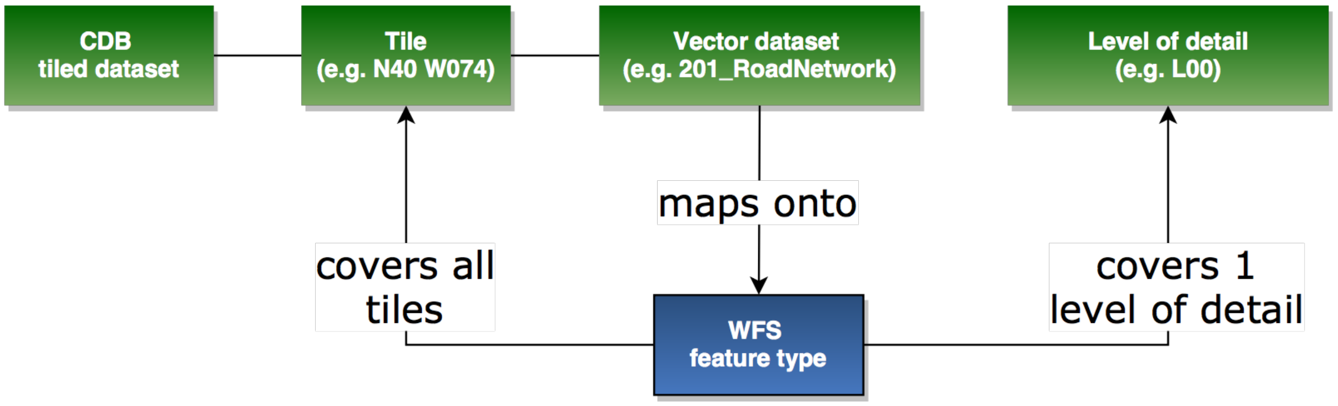


Figure 10. Organization of CDB tiled vector data into WFS feature types

Although the CDB tiling scheme is no longer available   as all tiles for a single level of detail are merged   WFS clients can use the OGC Filter to spatially filter the desired dataset into desired tiles.

To enable access to multiple levels of detail for a dataset, multiple approaches are possible:

- Model the levels of detail as separate feature types. The feature type’s name can reflect this by combining the name of the vector dataset and the covered level of detail. Although this approach is simple to model from the server-side, it does require a client to be knowledgeable about the intended scale ranges of each feature type. This could be mitigated by documenting intended scale ranges into the feature type’s metadata in the WFS capabilities document. However, there is no dedicated metadata element available for this in WFS 1.x or 2.0. For WFS clients supporting OGC’s styling language Symbology Encoding (SE) [http://www.opengeospatial.org/standards/se], a more convenient solution could be to define an SE that embeds the intended scale ranges into styling rules;
- Merge levels of detail into one feature type and enrich the data with an additional level of detail attribute. This approach is similar to the previous one, except that it shifts the level of detail aspect to the data itself. In this case, a WFS client will need to use an OGC Filter to filter the data according to the desired level of detail;
- Merge levels of detail into one feature type and include server-side heuristics to determine the right level of detail. For example, a server could rely on the geographical extent of the spatial filter to determine a suitable level of detail. If a WFS client sends a request covering 180 degrees by 90 degrees, the likelihood is that the lowest level of detail is necessary. Conversely, if the request includes a filter covering an area of 10 square miles, a high level of detail is likely to be needed. A benefit of this approach is the simplicity for the client, who only needs to request a single feature type to access a desired vector dataset. However, the heuristics might not always work. An example is the case in which a client requests data according to a client-side grid pattern, in which case, a spatial filter, etc. is not provided.

The first approach was chosen in the Testbed 13 CDB WFS implementation to serve multiple levels of *RoadNetwork* and *HydrographyNetwork* data from a CDB datastore.

An additional option to explicitly support CDB’s tiling schemes and multiple levels of detail is to extend the WFS protocol with additional parameters. For example, a _*GetFeature* operation could be overloaded with a tile identification number and a level of detail. However, to maintain WFS client/server interoperability, it is recommended to examine this at the standardization level before putting this into practice.

9.1.2. Serving Vector Data

A CDB-compliant datastore currently uses ShapeFiles for the representation and attribution of vector feature datasets. Aligned with the capabilities of the ShapeFile format, vector features consist of points, lines and polygon features. A few examples of CDB vector datasets include *road network_s, _railroad networks*, *powerline networks* and *hydrography networks*.

In a WFS environment, the ShapeFile data gets encoded according to a GML application schema that represents the structure of the data, which includes its properties and geometry. The appendices include an example of a GML application schema for a RoadNetwork dataset.

9.1.3. Serving 3D Model Data

To represent 3D models, CDB uses the OpenFlight and SGI Formats to represent the 3D geometries and textures respectively. In addition, the ShapeFile format is used to specify locations of the 3D models.

The OpenFlight and SGI formats are binary data formats and consequently are not compatible with a regular XML-based GML Application Schema used in a WFS environment. A few approaches can be identified to integrate the 3D data in a WFS environment:

- Embed the binary data in XML using Base64 encoding. A precedent in OGC standards can be found in the SE standard, which defines the use of this approach to embed binary bitmap data using icon styling rules. A drawback of this approach is the increased size of the data, given the fact that the ratio of output bytes to input bytes is 4:3 (33% overhead);
- Link to the original binary data using an additional property in the XML data. With this approach, the OpenFlight and SGI formatted data becomes accessible through a HTTP link encoded in a property of the data;
- Use a custom exchange format tailored to the use case. Although the WFS advocates the use of GML as an exchange format, it is possible to use other formats: the WFS capabilities include a list of supported output formats for each feature type, which can be used as a parameter by the WFS client in a *GetFeature* operation. Examples of candidate formats include, georeferenced *COLLADA* or *glTF* data, which are ideally suited for streaming of large amounts of 3D content and integrate well in web-based environments.

The second approach was chosen in the Testbed 13 CDB 3D WFS implementation to serve 3D vegetation data from a CDB datastore. A WFS feature type contains point features that specify the location and attributes of the vegetation data. The CDB 3D WFS adds one extra attribute to each feature, specifying an HTTP link to an OpenFlight 3D geometry model, which in turn links to SGI texture data relative to the OpenFlight data location. By downloading the OpenFlight 3D geometry model and the linked texture data, a WFS client is able to style the point features with 3D models of the vegetation.

The XML snippet below shows an example of one vegetation feature served by the Testbed 13 CDB 3D WFS. The property '*GTModel*' is introduced by the 3D WFS to link the feature to an OpenFlight 3D geometry model. The appendices include the corresponding GML application schema.

```xml
<?xml version='1.0' encoding='UTF-8'?>
<wfs:FeatureCollection xmlns:wfs="http://www.opengis.net/wfs/2.0"
                       xmlns="http://www.opengis.net/gml"
                       xmlns:cdb="vegetation/gml/3.2"
                       xmlns:gml="http://www.opengis.net/gml/3.2"
                       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                       timeStamp="2017-09-25T19:14:18.973Z">
  <wfs:boundedBy>
    <gml:boundedBy>
      <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
        <gml:lowerCorner>40.74056050300004 -73.97796738499994</gml:lowerCorner>
        <gml:upperCorner>40.74056050300004 -73.97796738499994</gml:upperCorner>
      </gml:Envelope>
    </gml:boundedBy>
  </wfs:boundedBy>
  <wfs:member>
    <cdb:vegetation gml:id="HC_1975342088">
      <cdb:geometry>
        <gml:Point gml:id="HC_1975342088_1">
          <gml:pos>40.74056050300004 -73.97796738499994</gml:pos>
        </gml:Point>
      </cdb:geometry>
      <cdb:id>0</cdb:id>
      <cdb:Latitude>N40</cdb:Latitude>
      <cdb:Longitude>W074</cdb:Longitude>
      <cdb:LOD>L00</cdb:LOD>
      <cdb:Up>U0</cdb:Up>
      <cdb:Right>R0</cdb:Right>
      <cdb:CNAM>N40W074L00U000R000~EC030~0000005</cdb:CNAM>
      <cdb:SCALx>1.19218</cdb:SCALx>
      <cdb:SCALy>1.19218</cdb:SCALy>
      <cdb:SCALz>1.19218</cdb:SCALz>
      <cdb:AO1>129.17</cdb:AO1>
      <cdb:RTAI>0.0</cdb:RTAI>
      <cdb:MODL>maple_medium_80ft</cdb:MODL>
      <cdb:FACC>EC030</cdb:FACC>
      <cdb:FSC>0.0</cdb:FSC>
      <cdb:BSR>21.119</cdb:BSR>
      <cdb:BBH>24.386</cdb:BBH>
      <cdb:BBW>24.386</cdb:BBW>
      <cdb:BBL>24.386</cdb:BBL>
      <cdb:HGT>24.39</cdb:HGT>
      <cdb:CMIX>1.0</cdb:CMIX>
      <cdb:AHGT>F</cdb:AHGT>
      <cdb:GTModel>

http://demo.luciad.com:8080/CDB/NewYork\GTModel\500_GTModelGeometry\E_Vegetation\C_Woo
dland\030_Trees/D500_S001_T001_EC030_000_maple_medium_80ft.flt
      </cdb:GTModel>
    </cdb:vegetation>
  </wfs:member>
</wfs:FeatureCollection>
```

9.2. Implementation

The CDB (3D) WFS was implemented and contributed by Luciad. The implementation builds on Luciad's COTS software product *LuciadFusion*, which offers standards-based software components designed for geospatial server development. The following sections discuss the service's functional and deployment characteristics.

9.2.1. Functional Overview

The Luciad CDB WFS provided the following functionality to support the CDB work in Testbed 13:

- Interface characteristics:
  - OGC-compliant WFS 1.1 & 2.0 service interface with support for the following requests: *GetCapabilities*, *DescribeFeatureType* and *GetFeature*. Supported request encodings are HTTP GET and POST;
  - Support for OGC Filter Encoding 1.1.0 & 2.0;
  - Use of GML as exchange format.
- Capabilities related to CDB:
  - Aggregation and delivery of CDB vector (Shapefile) data according to the following WFS feature type organization:
    - Multiple Levels of Detail (LoD) are mapped to separate feature types.
    - Each feature type consolidates all vector data available on the related LoD.
  - Data is served directly from the CDB source. Data throughput is optimized by streaming CDB vector data on a feature-by-feature basis. Additionally, caching is used to reduce the response time.

9.2.2. Deployment Characteristics

The Luciad CDB (3D) WFS implementation is based on Java Servlet technology. To run, the WFS requires a Java servlet container or application server compatible with Java Servlet 3.0 or higher. Apache Tomcat 8 was used by Luciad during Testbed 13. Other than being capable of running a Java Virtual Machine 1.8 (or higher) and an appropriate servlet container/application server, no other requirements are required of the underlying hardware or operating system.

Chapter 10. CDB WCS

The Web Coverage Service (WCS) [http://www.opengeospatial.org/standards/wcs] is an OGC standard interface for offering and accessing raster geospatial images as coverages. A coverage is geospatial information representing space-varying phenomena. Examples of coverages are satellite/aerial images, land cover data, digital elevation models and precipitation maps. Conceptually WCS is very similar to Web Map Service (WMS) [http://www.opengeospatial.org/standards/wms] which returns image formats. However, a WCS instance can return the actual coverage data such as metadata. The content can be returned to the requester using a multi-dimensional format such as NetCDF (Network Common Data Form). Therefore, WCS clients can extract a portion of the coverage that they need and run complex modeling and analysis in comparison to a WMS instance. A WCS instance can perform the following operations:

- The *GetCapabilities* operation is a request to a WCS server for a list of operations and services ("capabilities") offered by that server;
- The *DescribeCoverage* request returns additional information about the coverage such as its metadata, CRS, domain, range and formats. A client will generally need to issue a *DescribeCoverage* request to ensure it can make the proper *GetCoverage* request;
- The *GetCoverage* request retrieves the geospatial image data source as subsets of coverages or a reference to the coverage. The most powerful thing about a *GetCoverage* request is its ability to subset domains (height and time) and ranges. It can also do resampling, encode using different data formats, and return the resulting file in different ways.

The OGC WCS prototype for supporting CDB access offers multi-dimensional coverage data for access by a client request. This section discusses the current results and progress, and provides some background about the main components, the architecture, and examples of the interface along with explanations of the main client functions.

10.1. Development Plan and Architecture

The proposed cloud-based architecture for storing and manipulating CDB datastore is described below. In the following figure, a Java-based CDB API is designed and developed to read CDB datastores from cloud hosting and providers, and generate raster mosaics based on the CDB raster layers. After generation, the raster mosaics are published with WCS using a map-server (e.g. GeoServer). Map server WCS will access and publish each CDB coverage or raster data layer for consumption by WCS compatible clients. The proposed architecture for accessing the CDB using WCS is shown in Figure 11.

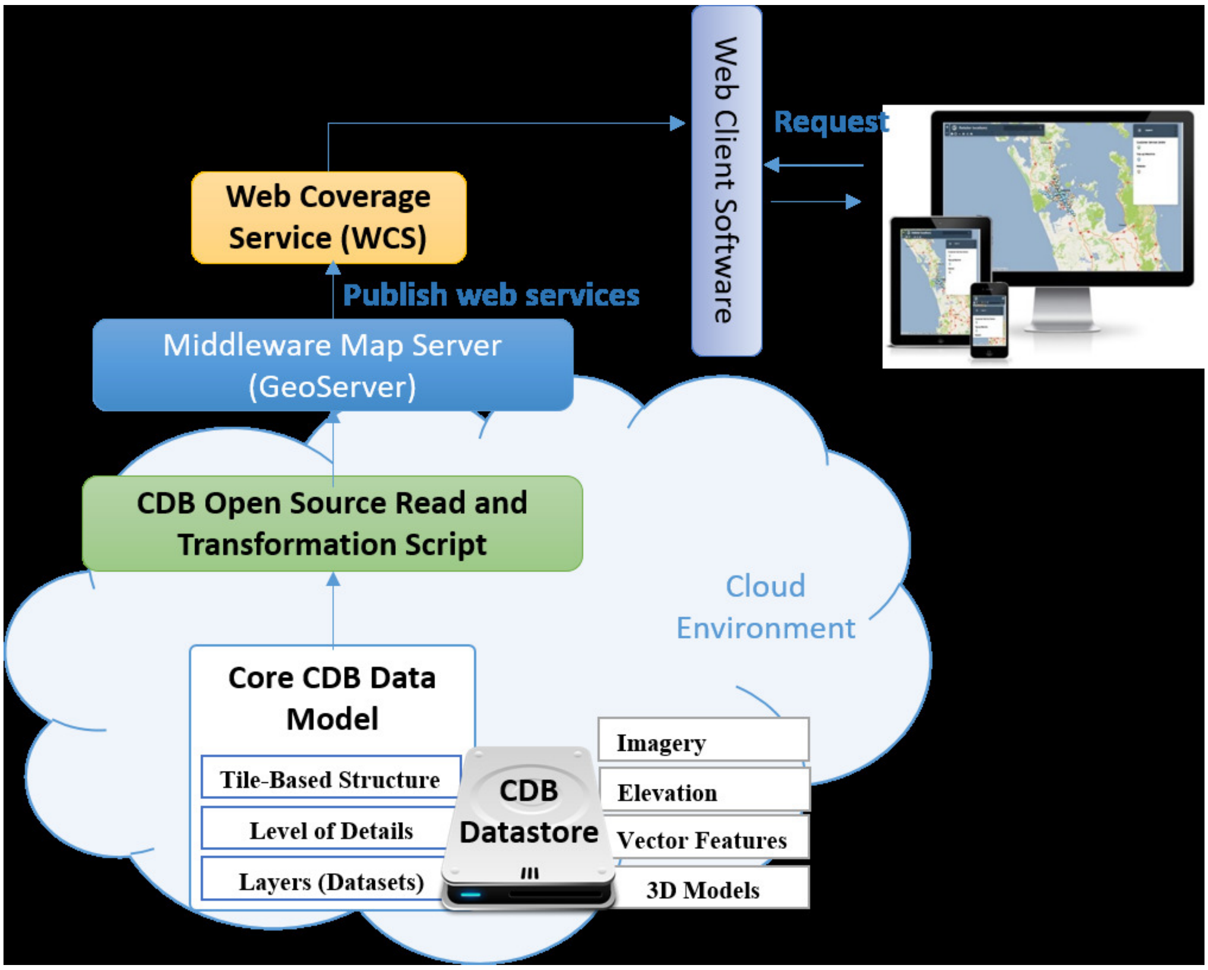


Figure 11. WCS CDB system architecture

10.2. WCS Component Design

WCS provides access to the raster datasets available in a CDB data store. The server can deliver data in the client's preferred format. The WCS components are described in Figure 12.

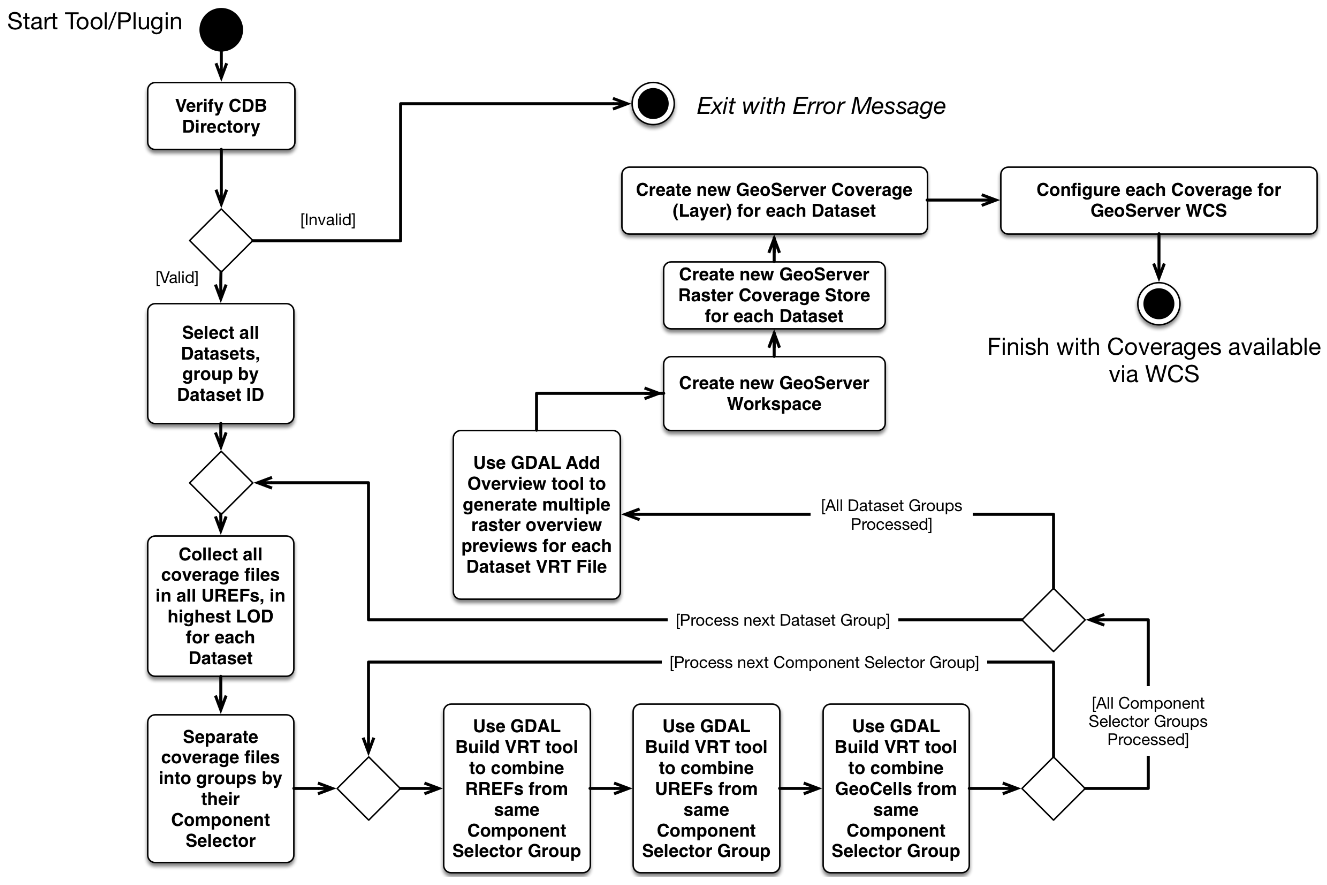

Start Tool/Plugin
Verify CDB Directory
[Invalid]
Exit with Error Message
[Valid]
Select all Datasets, group by Dataset ID
Collect all coverage files in all UREFs, in highest LOD for each Dataset
Separate coverage files into groups by their Component Selector
Use GDAL Build VRT tool to combine RREFs from same Component Selector Group
Use GDAL Build VRT tool to combine UREFs from same Component Selector Group
Use GDAL Build VRT tool to combine GeoCells from same Component Selector Group
[Process next Component Selector Group]
[All Component Selector Groups Processed]
[Process next Dataset Group]
[All Dataset Groups Processed]
Use GDAL Add Overview tool to generate multiple raster overview previews for each Dataset VRT File
Create new GeoServer Workspace
Create new GeoServer Raster Coverage Store for each Dataset
Create new GeoServer Coverage (Layer) for each Dataset
Configure each Coverage for GeoServer WCS
Finish with Coverages available via WCS

Figure 12. Component design and activity diagram of WCS process for CDB

10.2.1. CDB WCS Implementation for New York Dataset

For testing the WCS component, a NYC CDB sample dataset is provided for the Testbed 13 experiment. The University of Calgary's WCS server is online and accessible through the WCS 2.0.1 service as follows:

```
URL: https://gsw-app.gswlab.ca:8443/geoserver/wcs
username: tester
password: G45-DLB-Dcd-JaU
```

GeoServer supports WCS 1.0, 1.1.0, 1.1.1, and 2.0.1 and WCS should load in QGIS 2 or any other clients that support WCS. In QGIS, this layer can be added in the Browser panel under WCS by right-clicking on WCS and selecting "New Connection". Set any name and enter the above URL, username and password.

In the following figure (Figure 13), a snapshot of the CDB WCS is captured using the above URL and QGIS client.

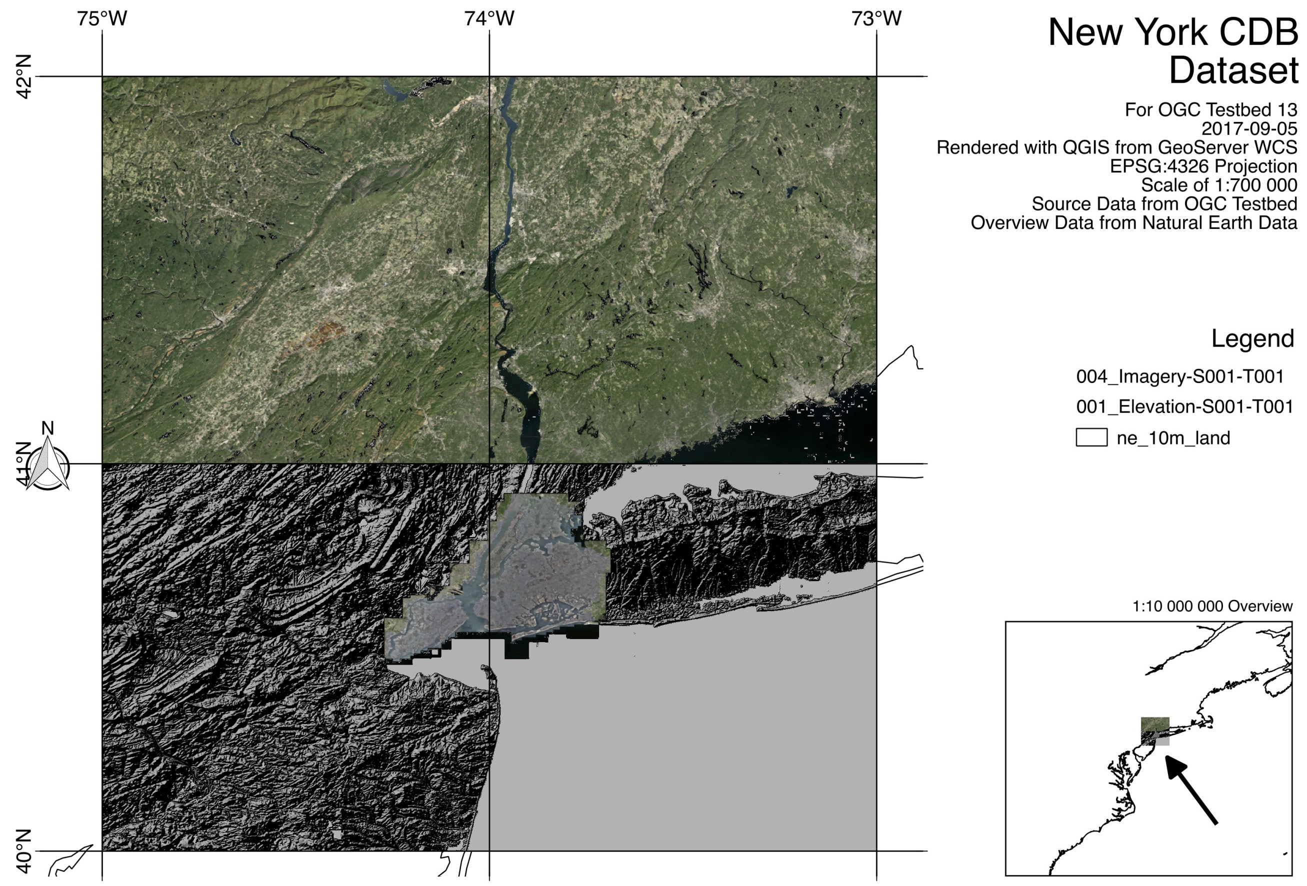


Figure 13. CDB WCS Preview NY Dataset

10.2.2. CDB WCS Implementation for New York Dataset (Interactive 3D Preview)

For testing the WCS component, the same Testbed 13 dataset is accessible through Cesium.js with a WCS elevation provider library. The client uses WebGL (Web Graphics Library) to render elevation and imagery data on a 3D globe in the web browser. Cesium.js only supports elevation data from WCS using a 3rd-party plugin, and Cesium.js does not support loading imagery data provided by a WCS instance at all. To address this shortcoming, the map server was configured to make the CDB imagery dataset available using WMS and WMTS instances, which are supported by Cesium.js. The Cesium.js demo can be viewed at:

```
URL: https://gsw-app.gswlab.ca/
username: tester
password: G45-DLB-Dcd-JaU
```

Note that a computer with a discrete GPU (Graphics Processing Unit) or a computer with an integrated high-performance GPU is recommended for smooth viewing performance. Also, a client certificate (available on the URL page [https://gsw-app.gswlab.ca/]) needs to be installed on the web browser to properly load authenticated data from WCS and WMTS.

In the following figure (Figure 14), a screenshot of the CDB WCS is captured using Cesium.js in the web browser.

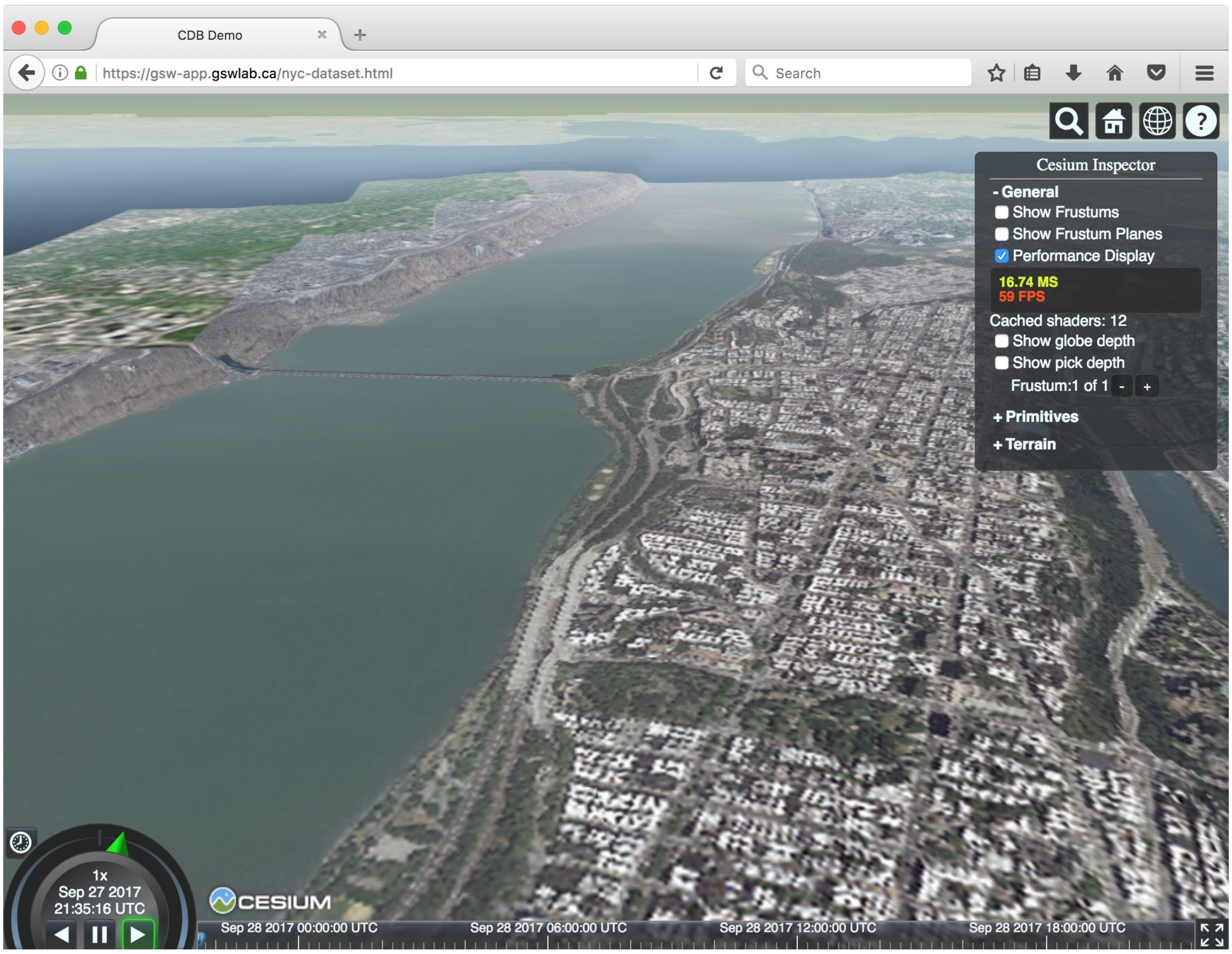


Figure 14. CDB WCS Cesium Preview

10.2.3. Performance Test with Overviews

When serving CDB raster data using WCS endpoints, the map server software may generate low resolution overviews on-the-fly if none are present in the source data. As CDB stores lower resolution tiles in separate directories (LoDs), the map server software must be configured to use LoDs for sending lower resolution overviews to the client. Such functionality will require custom plugins for the map server software to “understand” the CDB directory layout. However, no such plugin exists as yet for GeoServer. Due to this issue, GeoServer will create overviews on-the-fly resulting in slower performance for clients as they must wait for the server to generate overviews.

As an interim solution until a full CDB plugin is developed, GeoServer may use the GDAL plugin to read overviews embedded in raster files (or external overviews in separate files). With the Virtual Files (VRT) that point to high resolution CDB raster tiles, the GDAL Add Overview tool can be used to generate external overview files for each CDB Dataset VRT. The tool will not use lower resolution LoDs from CDB, it will instead read the high resolution tiles and generate multiple overview levels internally. Once these external overviews are available in the same directory as the VRTs, GeoServer with the GDAL plugin will automatically read the overviews and use them for the WCS and WMS requests.

The time taken to generate overviews using GDAL is dependent on the size of the dataset, with imagery spanning two geo-cells taking tens of minutes to generate overviews, and elevation data covering the same region taking a few minutes. This is a process that only needs to be run once whenever the dataset is updated, and the cost in time to generate and disk space is acceptable compared to the time saved for client download and rendering.

10.2.4. Performance Test with HTTP Caching

WCS requests are carried over HTTP and can take advantage of some of the caching features built into the protocol. Enabling server-side compression of text resources, especially XML documents such as Capability Documents, can significantly reduce the download time for clients that support compression encoding. Compression can be enabled by the application server so that the map server does not need to know about it, simplifying the map server configuration.

Other caching features such as ETags (Entity tag) or Modified-Since require information about the dataset and would have to be generated by the map server. GeoServer does not currently generate these for WCS requests, although it does for Web Map Tiles Service (WMTS) requests. These features will also require client support to avoid re-downloading assets.

10.2.5. Performance Test with HTTP/2

HTTP/2 is a successor to HTTP/1.1 that adds multiple features to reduce resource latency. Some of these features are compressing assets, request pipelining, multiplexing requests, and server push. HTTP/2 can be offered simultaneously with HTTP/1.1, so that clients that do not support the newer protocol can still access the HTTP server. For this project, the application server (Tomcat) is modified to enable HTTP/2 support for clients to access the OGC Web Services.

For testing with web browser clients such as Cesium.js, HTTP/2 was activated for demo assets and WCS requests. The loading performance for WCS tiles was improved as the browser did not have to wait for existing TCP connections to be freed to be able to download more tiles. The client, however, did not take advantage of HTTP caching, and would thus not re-use previously downloaded WCS tiles. This caused the client to delay loading the WCS as the data had to be re-downloaded every time.

10.2.6. Performance Test with WCS 2.0

OGC WCS 2.0 supports coverage encoding profiles which may improve loading performance for compatible clients. One of these profiles is for GeoTIFF, which allows for compression to be enabled in the GeoTIFF response body. During testing, coverages were downloaded using WCS 2.0.1 KVP (key-value pair) and then, the "&compression=DEFLATE" pair was added, as well as, compression schemes "PackBits" and "LZW". In the tests, compression reduced GeoTIFF request sizes by up to 75%. However the Cesium.js WCS library failed to parse the elevation data in the compressed files and would not load any elevation onto the demonstration. This is a client limitation.

10.2.7. Authentication Controls for WCS

HTTP Basic authentication was enabled on the map server to restrict access to CDB datasets to only individuals who have the username and password. These users are granted read-only access to the datasets. SSL is enabled for the application server to encrypt communications between the WCS and clients, otherwise the username and password would be sent in plain text over the network.

There is also X.509 client certificate authentication enabled on the application server and map server, but on a separate port from the HTTP Basic authentication setup. This was done to ensure that clients that do not support client certificates would still be able to access the WCS. Client certificate authentication was enabled as web clients may not be able to properly handle HTTP Basic for loading assets through image tags (e.g. Cesium.js). SSL is also enabled for client certificate users. There is currently only one read-only user certificate available, and can be downloaded from the Cesium.js demonstration website.

Chapter 11. CDB Client

The CDB client in Testbed 13 focused on two tasks:

- Interaction with the CDB web services - including an OGC WFS and OGC WCS - and visualizing the provided data.
- Transformation of CDB data to the NAS-based CDB Profile.

11.1. Architecture & Design

This section describes the architecture of the CDB client and the design decisions taken to support the envisioned use case. Figure 15 shows the high-level architecture of the client.

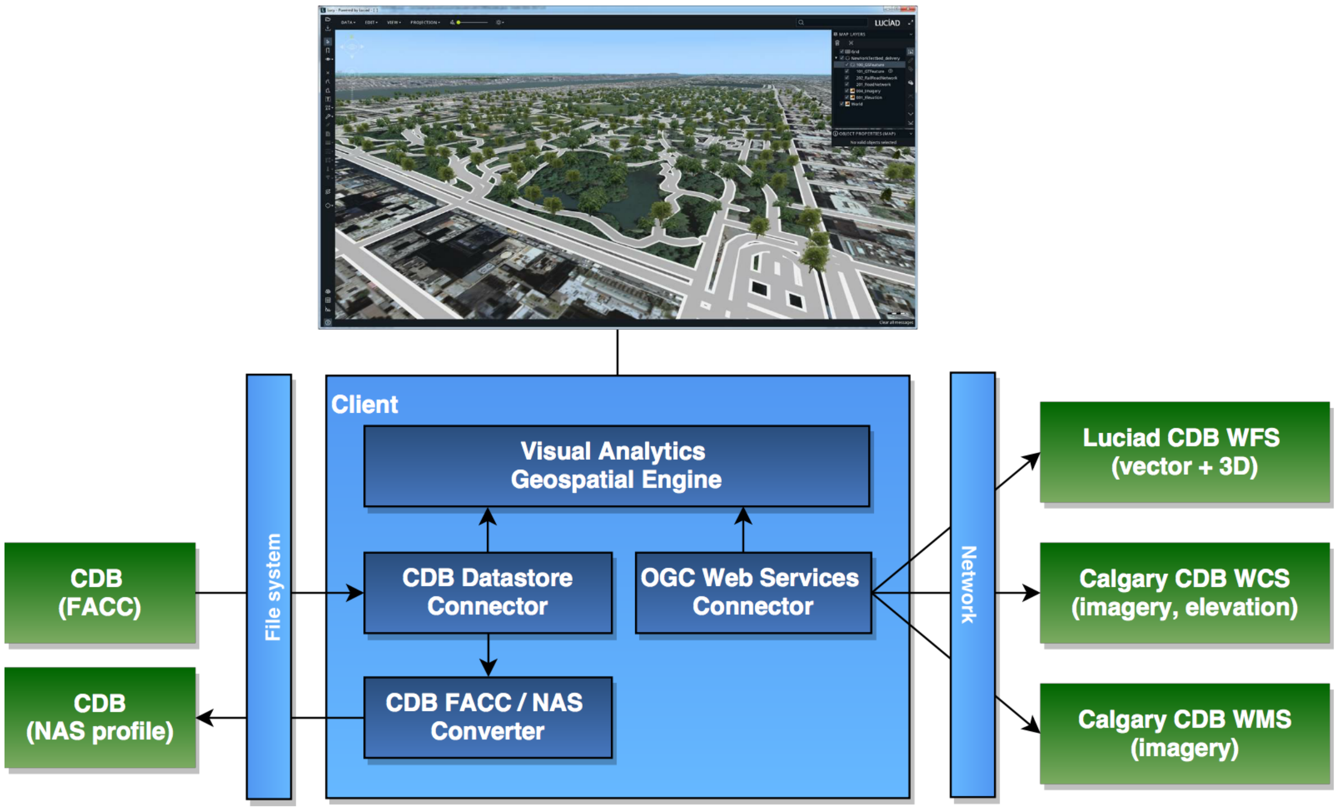

Client
Visual Analytics Geospatial Engine
CDB Datastore Connector
OGC Web Services Connector
CDB FACC / NAS Converter
File system
Network
CDB (FACC)
CDB (NAS profile)
Luciad CDB WFS (vector + 3D)
Calgary CDB WCS (imagery, elevation)
Calgary CDB WMS (imagery)

Figure 15. CDB client architecture

To support the loading of CDB data, the client can connect to two types of data sources:

- OGC web services, including:
  - An OGC WFS serving features types with vector & 3D data from a CDB datastore;
  - An OGC WCS serving coverages with imagery & elevation data from a CDB data store;
  - An OGC WMS serving layers with imagery from a CDB data store.
- Local CDB datastores accessible via the file system.

Additionally, the CDB client offers support to transform local CDB data to the NAS-based CDB Profile.

11.1.1. Handling CDB Content through OGC Web Services

The OGC CDB web services defined in Testbed 13 focus on interoperability with existing OGC web service clients, by adopting the default data exchange formats - GeoTIFF for WCS, a GML application schema for WFS, JPEG / PNG for WMS - and by avoiding any vendor-specific extensions on top of the web service protocols. Consequently, it is easy for a general-purpose OGC web service client to interact with the CDB web services and consume the majority of the CDB content.

The only remaining difficulty is in accessing the 3D model data. This requires additional support from the client to visualize the data as expected. As discussed in the CDB WFS Chapter, the 3D model data is represented by a WFS feature type per data set and level of detail, in which each feature contains the attributes and location of the corresponding 3D model. Aligned with the CDB format, the actual 3D model is an OpenFlight geometry model that links to textures in the SGI format. This 3D model is made accessible via an HTTP link inside one of the WFS feature's attributes. Consequently, a WFS client needs to support the OpenFlight and SGI formats and resolve the HTTP link to load the corresponding OpenFlight / SGI data and to render the data as expected.

To ease the process of finding the right attribute, an OGC SE styling file could be defined that defines a point symbolizer for the data with an external graphic that links to the attribute containing the OpenFlight / SGI data link. The following XML snippet contains an example of an OGC SE point symbolizer for the 3D vegetation data offered by the CDB WFS. The location of the icon - i.e. the OnlineResource href attribute - relies on the GTModel feature attribute, which contains the HTTP link to the 3D model.

```
<?xml version='1.0' encoding='UTF-8'?>
<PointSymbolizer>
  <Graphic>
    <ExternalGraphic>
      <OnlineResource xlink:href="${GTModel}"/>
      <Format>application/vnd.openflight</Format>
    </ExternalGraphic>
  </Graphic>
</PointSymbolizer>
```

11.1.2. Converting FACC-based CDB data to the NAS Profile

Next to consuming CDB data, the CDB client in Testbed 13 is also able to convert a local CDB datastore to the NAS Profile. This includes the mapping of FACC to NAS codes to identify the data inside a CDB datastore - for instance, the FACC code for vegetation is EC030 while the NAS code is EC005. This mapping affects the resources of a CDB datastore that can include a FACC code, including:

- .dbf files: .dbf files exposing FACC information include a FACC column with the corresponding code;
- .xml metadata files: the .xml metadata files include a <Feature_Attribute_Catalog_Code> metadata entry exposing the FACC code;
- directory names: the FACC code can be used to orgainze the data in the CDB directory structure.

An example for trees is E_Vegetation\C_Woodland\030_Trees inside a GTModel.

In the above resources, the conversion logic can simply replace the code with the NAS equivalent - this is the implementation approach used by the CDB client in Testbed 13. Alternatively, the semantics of any involved property might also be changed to emphasize the adoption NAS profile: in the above resources, this could mean the addition of a NAS column to the .dbf files and the integration of a <NAS> metadata entry in the .xml metadata files.

11.2. Implementation

The CDB client was implemented and contributed by Luciad. The implementation builds on top of Luciad's COTS software product *Luciad Lightspeed*, which offers standards-based software components designed for geospatial application development. The following sections discuss the client's functional and deployment characteristics and include a few screenshots to illustrate the client's HMI and integration with the New York CDB data source.

11.2.1. Functional Overview

The Luciad CDB client provides the following functionality to support the CDB work in Testbed 13:

- Map-centric 2D & 3D display with an intuitive user interface giving access to various actions, including:
  - Connecting to and loading of data sources (files, databases, web services …)
  - Map interactions (data layer control, zoom, pan, selection, feature information access)
  - Map controllers to manipulate the map (zoom, pan, …)
  - Geocoding functionality to quickly search for & navigate to places
- Client interface to OGC web services:
  - WFS-T 1.0, 1.1 & 2.0, including support for Filter Encoding 1.0, 1.1 and 2.0
  - FPS/WMS 1.1.1 & 1.3.0, including support for SLD 1.0 / 1.1.
  - WCS 1.0 & 1.1.
  - WMTS 1.0.
  - WPS 1.0 & 2.0.
  - CSW 2.0 & 3.0.
- Wide range of data format support, including (non-exhaustive):
  - OGC CDB 1.0
  - 3D format support (CityGML, OBJ, OpenFlight …) for data such as 3D terrain and 3D buildings.
  - Imagery formats: GeoTIFF, TIFF, JPEG, JPEG 2000, PNG, GIF, ECW, MrSID, CADRG / ADRG / USRP, NITF, ERDAS Imagine.
  - Weather / gridded data formats: WXXM 1.1 / 2.0, GRIB 1 & 2, NetCDF 1.0.
  - Terrain data formats: DMED / DTED, USGS DEM, Swiss DHM, GeoTIFF.

    - Vector formats: ESRI Shape, MapInfo MIF/MAP, NVG, GML 2/3.1.1/3.2.1, GeoJSON.
    - Other: OGC KML 2.2, OGC GeoPackage 1.0, GeoPDF.
- Transformation of CDB data to the NAS-based CDB Profile.

11.2.2. Deployment Characteristics

All development was done in Java, using the Java Development Kit (JDK) 1.8 and Luciad's COTS product *Luciad Lightspeed*. The software runs on any operating system for which a Java Virtual Machine 1.8 or higher exists. For the 3D visualization, a graphics card with 512MB RAM and support for OpenGL 2.0 or higher is required.

11.2.3. New York CDB Data Integration Results

The following screenshots show the CDB client interacting with the CDB dataset for New York. This includes the visualization of imagery, elevation, vector and 3D data available in the CDB dataset.

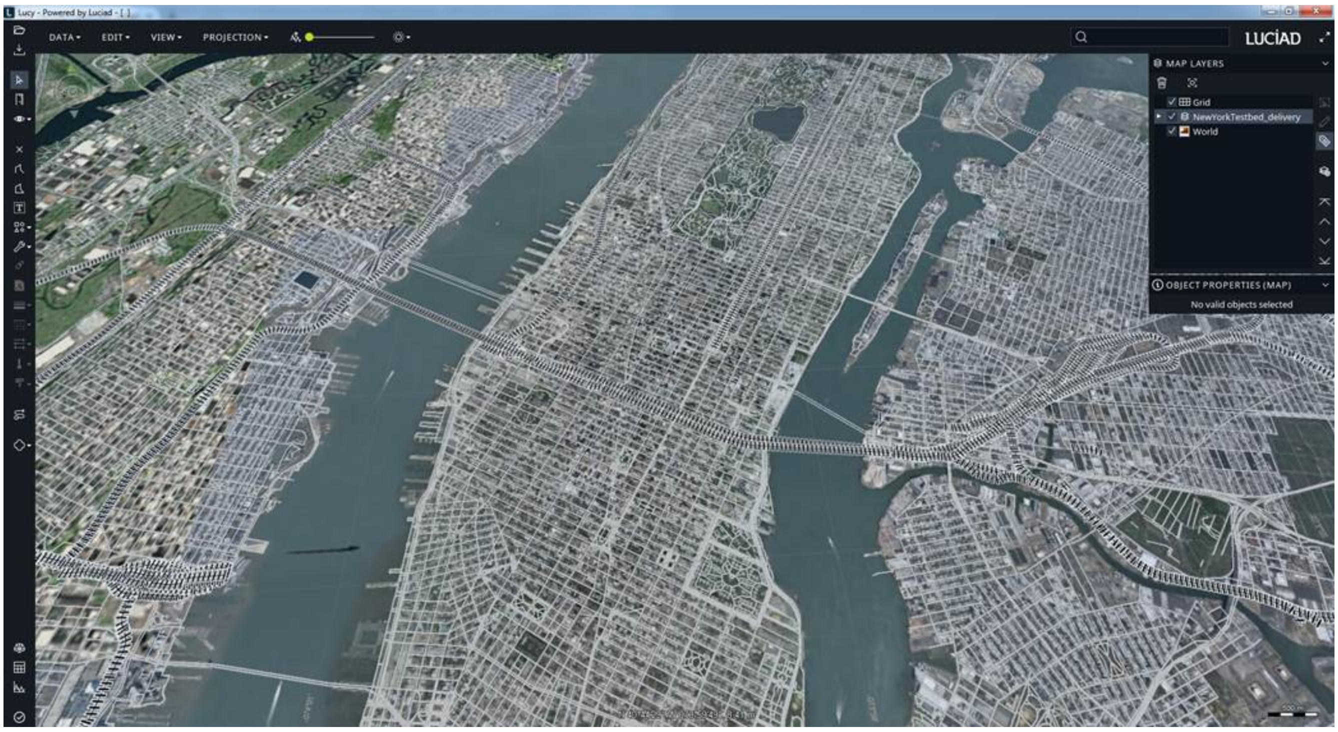


Figure 16. Luciad CDB client showing imagery, roads and rails around New York.

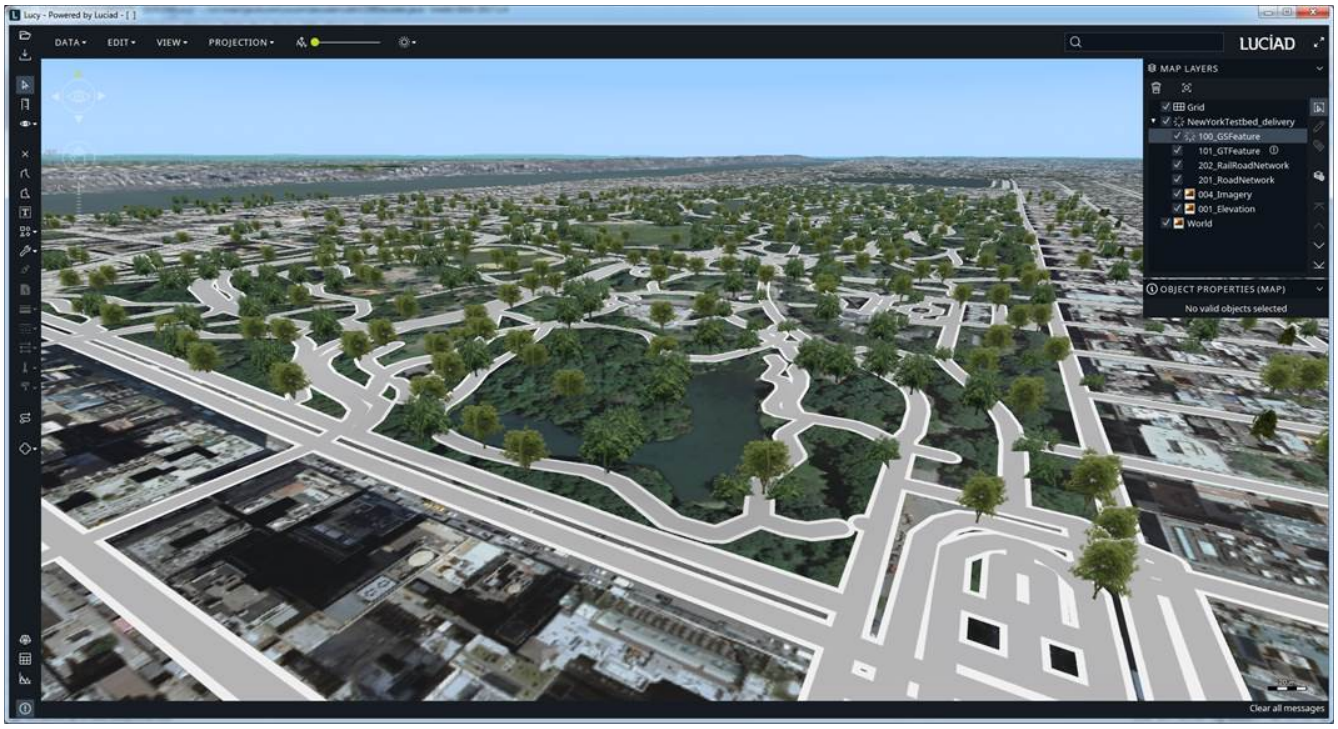


Figure 17. Luciad CDB client showing imagery, elevation, roads and 3D trees around New York.

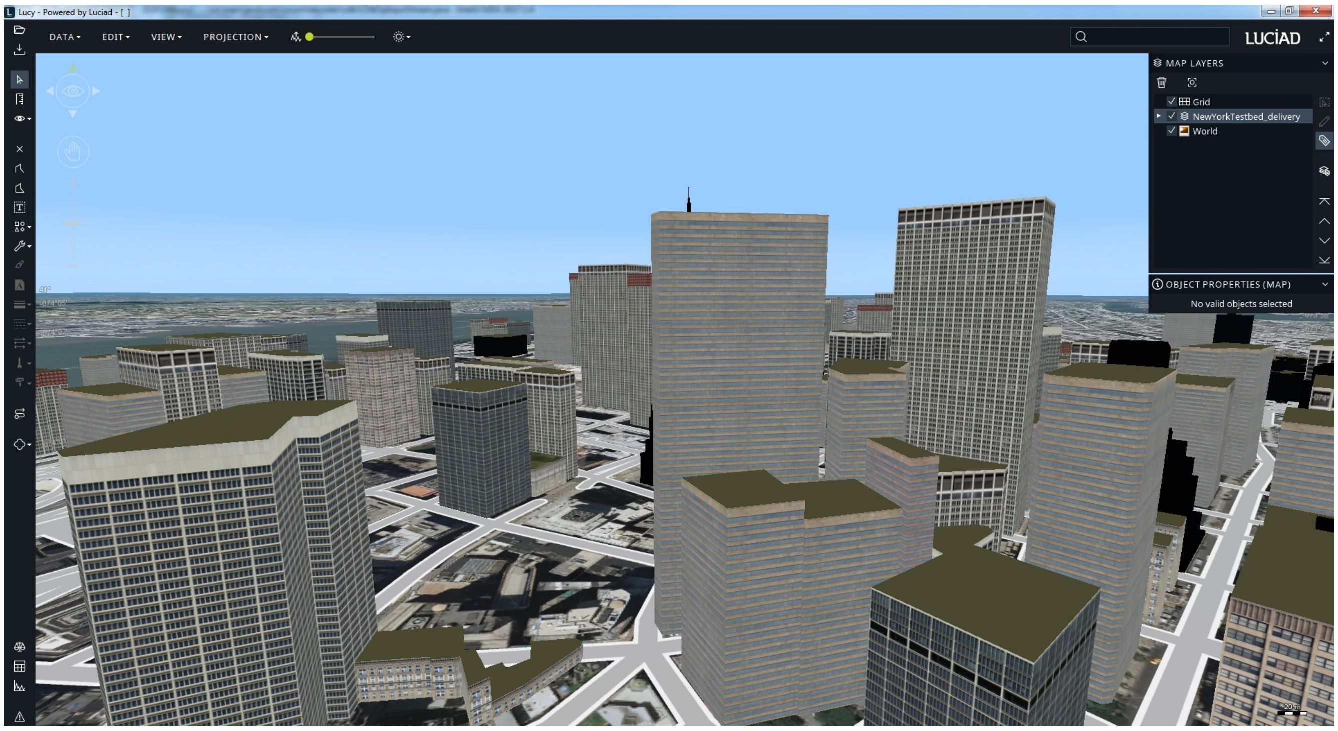


Figure 18. Luciad CDB client showing imagery, roads and 3D buildings around New York.

11.3. Key Findings and Recommendations

The work and experiments that were done in the OGC Testbed-13 CDB sub-thread are described in this CDB ER document. This ER was reported to the CDB SWG on several occasions and the current key findings of the CDB sub-thread are summarized here:

- The feasibility study for extending OGC CDB 1.0 feature codes and attributes reveals that the current version (1.0) supports only a limited extension:
    - The feature type codes which existed in the FDD.xml files are a fixed set, and the extension

    needs hard coding of the newly added codes;

    ◦ CDB has an indexing structure (based on FACC codes) to save feature types inside the folder hierarchy. This indexing is also a limitation for defining the feature codes. If an arbitrary alphabetical feature code is used inside the CDB; then, the indexing mechanism will be broken. Another approach is to define a mapping between the new alphabetical feature codes and old indexing;

    ◦ The attributes schema is defined for the tiled vectors inside the Shapefile database file. However, it can be extended in the current version of the CDB using extended attributes.

- The OGC CDB 1.0 can be accessed by OGC web services which support a NAS profile in organizing features and attributes and indexing vector features
- Change Requests have been discussed for CDB 2.0 and will be submitted:
    - The recommendations for replacing FACC feature code and indexing structure to be consistent with the application schemas (*e.g.* NAS data model) under discussion in CDB 2.0;
    - The recommendations for supporting application schemas in CDB (level of complexity: Esri Geodatabase, GML-SF0 application schema) are being discussed for CDB 2.0.
    - The method to expand the supported encodings and formats for the CDB datastore
    - Generating a coherent attribute schema for CDB 1.0 based on the "CDB_Attribute.xml" file

The following recommendations are envisioned for future research to improve upon results of this ER:

1. A performance analysis and evaluation for using CDB extended with NAS profile in run-time simulations;
2. Describe explicitly how the CDB model may or may not align with the other OGC baseline and standards (*e.g*, SOS, DGGS standard);
3. Extend the supported encodings and formats for a CDB database to include the use of the OGC GeoPackage, CityGML, LandInfra and IndoorGML standards as well as other broadly used community encoding standards, such as COLLADA. This work may require performing OGC interoperability experiments to understand the implications of these options better.
4. Consider replacement(s) for the feature data dictionary with the application schemas (*e.g.*, NAS) which described the features and attributes in the different domain of application with their associations and relationships.

Appendix A: WFS GML Application Schema Documents

This appendix includes two examples of WFS GML Application Schema documents that have been used in the OGC Testbed 13 to serve vector features and 3D data via a WFS.

1.1. RoadNetwork GML Application Schema

```
<?xml version='1.0' encoding='UTF-8'?>
<xsd:schema xmlns:tns="DataModel_201_RoadNetwork/gml/3.2" xmlns:xsd=
"http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml/3.2"
targetNamespace="DataModel_201_RoadNetwork/gml/3.2" elementFormDefault="qualified"
version="1.0">
  <xsd:import namespace="http://www.opengis.net/gml/3.2" schemaLocation=
"http://demo.luciad.com:8080/OgcCdbServices/schemas/gml/3.2.1/gml.xsd"/>
  <xsd:element name="FeatureCollection" type="tns:FeatureCollectionType"
substitutionGroup="gml:AbstractGML"/>
  <xsd:element name="DataModel_201_RoadNetwork_Data" type=
"tns:DataModel_201_RoadNetwork_DataType" substitutionGroup="gml:AbstractFeature"/>
  <xsd:complexType name="FeatureCollectionType">
    <xsd:complexContent>
      <xsd:extension base="gml:AbstractFeatureType">
        <xsd:sequence>
          <xsd:element name="featureMember" minOccurs="0" maxOccurs="unbounded">
            <xsd:complexType>
              <xsd:complexContent>
                <xsd:extension base="gml:AbstractFeatureMemberType">
                  <xsd:sequence>
                    <xsd:element ref="gml:AbstractFeature"/>
                  </xsd:sequence>
                </xsd:extension>
              </xsd:complexContent>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="DataModel_201_RoadNetwork_DataPropertyType">
    <xsd:sequence>
      <xsd:element ref="tns:DataModel_201_RoadNetwork_Data" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="gml:AssociationAttributeGroup"/>
    <xsd:attributeGroup ref="gml:OwnershipAttributeGroup"/>
  </xsd:complexType>
  <xsd:complexType name="DataModel_201_RoadNetwork_DataType">
    <xsd:complexContent>
```

```
      <xsd:extension base="gml:AbstractFeatureType">
        <xsd:sequence>
          <xsd:element name="id" minOccurs="0" type="xsd:long"/>
          <xsd:element name="Latitude" minOccurs="0" type="xsd:string"/>
          <xsd:element name="Longitude" minOccurs="0" type="xsd:string"/>
          <xsd:element name="LOD" minOccurs="0" type="xsd:string"/>
          <xsd:element name="Up" minOccurs="0" type="xsd:string"/>
          <xsd:element name="Right" minOccurs="0" type="xsd:string"/>
          <xsd:element name="DataModel_201_RoadNetwork_Geometry" minOccurs="0" type=
"gml:MultiGeometryPropertyType"/>
          <xsd:element name="CNAM" minOccurs="0" type="xsd:string"/>
          <xsd:element name="FACC" minOccurs="0" type="xsd:string"/>
          <xsd:element name="FSC" minOccurs="0" type="xsd:double"/>
          <xsd:element name="CMIX" minOccurs="0" type="xsd:double"/>
          <xsd:element name="TRF" minOccurs="0" type="xsd:double"/>
          <xsd:element name="LTN" minOccurs="0" type="xsd:double"/>
          <xsd:element name="HGT" minOccurs="0" type="xsd:double"/>
          <xsd:element name="DIR" minOccurs="0" type="xsd:double"/>
          <xsd:element name="AHGT" minOccurs="0" type="xsd:string"/>
          <xsd:element name="LENL" minOccurs="0" type="xsd:double"/>
          <xsd:element name="WGP" minOccurs="0" type="xsd:double"/>
          <xsd:element name="RTAI" minOccurs="0" type="xsd:double"/>
          <xsd:element name="SJID" minOccurs="0" type="xsd:string"/>
          <xsd:element name="EJID" minOccurs="0" type="xsd:string"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

1.2. 3D Vegetation GML Application Schema

```
<?xml version='1.0' encoding='UTF-8'?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns="
vegetation/gml/3.2" xmlns:gml="http://www.opengis.net/gml/3.2" targetNamespace=
"vegetation/gml/3.2" elementFormDefault="qualified" version="1.0">
  <xsd:import namespace="http://www.opengis.net/gml/3.2" schemaLocation=
"http://demo.luciad.com:8080/OgcCdbServices/schemas/gml/3.2.1/gml.xsd"/>
  <xsd:element name="FeatureCollection" type="tns:FeatureCollectionType"
substitutionGroup="gml:AbstractGML"/>
  <xsd:element name="vegetation" type="tns:vegetationType" substitutionGroup=
"tns:AbstractFeature"/>
  <xsd:element name="AbstractFeature" type="tns:AbstractFeatureType"
substitutionGroup="gml:AbstractFeature"/>
  <xsd:complexType name="FeatureCollectionType">
    <xsd:complexContent>
      <xsd:extension base="gml:AbstractFeatureType">
        <xsd:sequence>
          <xsd:element name="featureMember" minOccurs="0" maxOccurs="unbounded">
            <xsd:complexType>
              <xsd:complexContent>
                <xsd:extension base="gml:AbstractFeatureMemberType">
                  <xsd:sequence>
                    <xsd:element ref="gml:AbstractFeature"/>
                  </xsd:sequence>
                </xsd:extension>
              </xsd:complexContent>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="vegetationPropertyType">
    <xsd:sequence>
      <xsd:element ref="tns:vegetation" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="gml:AssociationAttributeGroup"/>
    <xsd:attributeGroup ref="gml:OwnershipAttributeGroup"/>
  </xsd:complexType>
  <xsd:complexType name="AbstractFeatureType" abstract="true">
    <xsd:complexContent>
      <xsd:extension base="gml:AbstractFeatureType"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="vegetationType">
    <xsd:complexContent>
      <xsd:extension base="tns:AbstractFeatureType">
        <xsd:sequence>
          <xsd:element name="geometry" minOccurs="0" type="gml:PointPropertyType"/>
          <xsd:element name="id" minOccurs="0" type="xsd:int"/>
          <xsd:element name="Latitude" minOccurs="0" type="xsd:string"/>
          <xsd:element name="Longitude" minOccurs="0" type="xsd:string"/>
          <xsd:element name="LOD" minOccurs="0" type="xsd:string"/>
          <xsd:element name="Up" minOccurs="0" type="xsd:string"/>
          <xsd:element name="Right" minOccurs="0" type="xsd:string"/>
          <xsd:element name="CNAM" minOccurs="0" type="xsd:string"/>
          <xsd:element name="SCALx" minOccurs="0" type="xsd:double"/>
          <xsd:element name="SCALy" minOccurs="0" type="xsd:double"/>
          <xsd:element name="SCALz" minOccurs="0" type="xsd:double"/>
          <xsd:element name="AO1" minOccurs="0" type="xsd:double"/>
          <xsd:element name="RTAI" minOccurs="0" type="xsd:double"/>
          <xsd:element name="MODL" minOccurs="0" type="xsd:string"/>
          <xsd:element name="FACC" minOccurs="0" type="xsd:string"/>
          <xsd:element name="FSC" minOccurs="0" type="xsd:double"/>
          <xsd:element name="BSR" minOccurs="0" type="xsd:double"/>
          <xsd:element name="BBH" minOccurs="0" type="xsd:double"/>
          <xsd:element name="BBW" minOccurs="0" type="xsd:double"/>
          <xsd:element name="BBL" minOccurs="0" type="xsd:double"/>
          <xsd:element name="HGT" minOccurs="0" type="xsd:double"/>
```

```
          <xsd:element name="CMIX" minOccurs="0" type="xsd:double"/>
          <xsd:element name="AHGT" minOccurs="0" type="xsd:string"/>
          <xsd:element name="GTModel" minOccurs="0" type="xsd:string"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="AbstractFeaturePropertyType">
    <xsd:sequence>
      <xsd:element ref="tns:AbstractFeature" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attributeGroup ref="gml:AssociationAttributeGroup"/>
    <xsd:attributeGroup ref="gml:OwnershipAttributeGroup"/>
  </xsd:complexType>
</xsd:schema>
```

Appendix B: Revision History

Table 13. Revision History

Date	**Release**	**Editor**	**Primary clauses modified**	**Descriptions**
June 30, 2017	0.1	Sara Saeedi	all	CDB Initial Engineering Reports (IER)
July 31, 2017	0.9	Sara Saeedi	all	Comments, scenario, architecture and content integrated
September 30, 2017	1.0	Sara Saeedi	all	CDB Draft Engineering Reports (DER)
October 16, 2017	1.1	Sara Saeedi	all	Adding the comments from CDB sub-thread review
October 30, 2017	1.2	Terry Idol	all	Editing and revising the CDB DER
November 1, 2017	1.3	Sara Saeedi	all	Final edits before submitting the CDB DER to the pending document
November 15, 2017	2	Sara Saeedi	all	First round of edits from the CDB SWG

Appendix C: Bibliography

[1] NN: OGC CDB Core Standard, Feature Data Dictionary, Version 1.0, http://schemas.opengis.net/cdb/1.0/Metadata/Feature_Data_Dictionary.xml

[2] NN: OGC CDB Core Standard, CDB Attributes, Version 1.0, http://schemas.opengis.net/cdb/1.0/Metadata/CDB_Attributes.xml

[3] Echterhoff, J. and Portele, C. (Editors): OGC Testbed-13: NAS Profiling Engineering Report, Work in progress, https://portal.opengeospatial.org/wiki/pub/Testbed13/ConvertDocsT13Output/T13/NG003-NASProfiling.html, to be published at http://docs.opengeospatial.org/per/17-???.html

[4] Saeedi, S.; Liang, S.; Graham, D.; Lokuta, M. F. and Mostafavi, M.A.: Overview of the OGC CDB Standard for 3D Synthetic Environment Simulation & Modeling. ISPRS International Journal of Geo-Information. 6(10), 306 (2017)

[5] Saeedi, S. (Editor): CDB to NAS Mapping for Feature Codes and Attributes, Work in progress, https://docs.google.com/document/d/1PduilxjxVFrue8aJ-urCn931idtZhYLDAccsVfCx-UU/edit?usp=sharing; and, will be published as an OGC document 17-082.

[6] Common Database (CDB) Specification, Release Notes Version 3.2, Update 1, 22 February 2016, http://external.opengeospatial.org/twiki_public/pub/CDBswg/WebHome/CDB_Specification_-_Release_Notes.pdf

[7] Graham, D. (Editor): OGC Common DataBase Volume 1 Main Body, OGC Best Practice Document, (2015) https://portal.opengeospatial.org/files/?artifact_id=61935

[8] Graham, D. (Editor): OGC Common DataBase Volume 2 Appendices, OGC Best Practice Document, (2015) https://portal.opengeospatial.org/files/?artifact_id=61936

[9] Reed, C. (Editor): Volume 1: OGC CDB Core Standard: Model and Physical Data Store Structure, Version 1.0, http://www.opengeospatial.org/standards/cdb

[10] Saeedi, S. (Editor): Volume 11: OGC CDB Core Standard Conceptual Model, Version 1.0, http://www.opengeospatial.org/standards/cdb