Volume 10

OGC CDB Implementation Guidance (Best Practice)

Table of Contents

Open Geospatial Consortium

Submission Date: 2020-01-21

Approval Date: 2020-xx-xx

Publication Date: 2020-xx-xx

External identifier of this OGC® document: http://www.opengis.net/doc/BP/cdb-implementation-guidance/1.2

Additional Formats (informative):

Internal reference number of this OGC® document: 16-006r5

Version: 1.2

Category: OGC® Best Practice

Editor: Carl Reed

Volume 10: OGC CDB Implementation Guidance (Best Practice)

Warning

This document defines an OGC Best Practices on a particular technology or approach related to an OGC standard. This document is **not** an OGC Standard and may not be referred to as an OGC Standard. It is subject to change without notice. However, this document is an **official** position of the OGC membership on this particular technology topic.

Document type:	Draft OGC® Best Practice
Document subtype:	
Document stage:	Candidate
Document language:	English

License Agreement

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

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

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

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

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

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize

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

i. Abstract

This document provides detailed implementation guidance for developing and maintaining a CDB compliant data store.

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

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

The CDB standard defines an open format for the storage, access and modification of a synthetic environment database. A **synthetic environment** is a computer simulation that represents activities at a high level of realism, from simulation of theaters of war to factories and manufacturing processes. These environments may be created within a single computer or a vast distributed network connected by local and wide area networks and augmented by super-realistic special effects and accurate behavioral models. SE allows visualization of and immersion into the environment being simulated [2: "Department of Defense Modeling and Simulation (M&S) Glossary", DoD 5000.59-M,].

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

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

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

and files with internal (industry-standard) formats.

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

ii. Keywords

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

ogcdoc, OGC document, ogcdoc, cdb, implementation guidance, simulation, synthetic environment

iii. Preface

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

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

iv. Submitting organizations

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

Organization name(s)

- CAE Inc.
- Carl Reed, OGC Individual Member
- Envitia, Ltd
- Glen Johnson, OGC Individual Member
- KaDSci, LLC
- Laval University
- Open Site Plan
- University of Calgary
- UK Met Office

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

v. Submitters

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

Name	Affiliation
Carl Reed	Carl Reed & Associates
David Graham	CAE Inc.

vi. Document Organization

include::https://github.com/opengeospatial/cdb-volume-1/blob/master/list_For ease of editing and review, the standard has been separated into 16 Volumes, one being a schema repository.

- Volume 0: OGC CDB Companion Primer for the CDB standard (Best Practice).
- Volume 1: OGC CDB Core Standard: Model and Physical Data Store Structure. The main body (core) of the CDB standard (Normative).
- Volume 2: OGC CDB Core Model and Physical Structure Annexes (Best Practice).
- Volume 3: OGC CDB Terms and Definitions (Normative).
- Volume 4: OGC CDB Rules for Encoding CDB Vector Data using Shapefiles (Best Practice).
- Volume 5: OGC CDB Radar Cross Section (RCS) Models (Best Practice).
- Volume 6: OGC CDB Rules for Encoding CDB Models using OpenFlight (Best Practice).
- Volume 7: OGC CDB Data Model Guidance (Best Practice).
- Volume 8: OGC CDB Spatial Reference System Guidance (Best Practice).
- Volume 9: OGC CDB Schema Package: http://schemas.opengis.net/cdb/ provides the normative schemas for key features types required in the synthetic modelling environment. Essentially, these schemas are designed to enable semantic interoperability within the simulation context (Normative).
- Volume 10: OGC CDB Implementation Guidance (Best Practice).
- Volume 11: OGC CDB Core Standard Conceptual Model (Normative).
- Volume 12: OGC CDB Navaids Attribution and Navaids Attribution Enumeration Values (Best Practice).
- Volume 13: OGC CDB Rules for Encoding CDB Vector Data using GeoPackage (Normative, Optional Extension).
- Volume 14: OGC CDB Guidance on Conversion of CDB Shapefiles into CDB GeoPackages (Best Practice).
- Volume 15: OGC CDB Optional Multi-Spectral Imagery Extension (Normative). of_volumes.adoc[]

vii. Future Work

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

The OGC CDB Standards Working Group (SWG) members understand there is a requirement for eventual alignment of the CDB standard with the OGC/ISO standards baseline. In Version 1 of the

CDB standard, effort was invested to begin aligning terminology and concepts, specifically in the coordinate reference system discussions and requirements.

The current version of the CDB standard is fully backwards compatible with version 3.2 of the CDB specification as defined and implemented by the current CDB implementer and user community. The requirements for a CDB data store are focused on the ability to store, manage, and access extremely large volumes of geographic content. In this version of the standard, initial harmonization with the OGC and ISO standards baseline has begun. For example, where appropriate, the CDB simulation community terms and definitions have been replaced with OGC/ISO terms and definitions. Further, the standards documents have been reorganized and structured to be consistent with the OGC Modular Specification Policy. However, the CDB SWG and community recognize the need to further harmonize and align this standard with the OGC baseline and other IT best practices. There has already been considerable discussion in this regard.

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

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

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

Chapter 1. Scope

This document provides detailed implementation guidance for developing and maintaining a CDB compliant data store.

Chapter 2. Conformance

Not Applicable

Chapter 3. References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

- Volume 1: OGC CDB Core Standard: Model and Physical Data Store Structure. The main body (core) of the CBD standard (Normative).
- Volume 2: OGC CDB Core Model and Physical Structure Annexes (Best Practice).

Chapter 4. Terms and Definitions

For the purposes of this document, the following abbreviations apply. All other terms and definitions are contained in Volume 3: OGC CDB Terms and Definitions (http://www.opengeospatial.org/standards/cdb).

RTP: Run-Time Publishers **SE:** Synthetic Environment

Chapter 5. Platform Recommendations

This section provides recommendations for platform minimum capabilities to implement a CDB Data Store. The platform constraints imposed by the CDB standard are minimal and are designed to allow implementation in many of the widely available computer hardware platforms, operating systems, file systems and transport protocols. Below are the suggested Platform requirements.

5.1. File System

A CDB data store instance is file system independent, (i.e., the use of a specific file system is not specified). However, implementation of the CDB standard does require that the file system be able to support a minimal set of capabilities as listed below:

1. File name/Directory name structure:
    1. Character set: in accordance to with Decimal, Hexadecimal, and Character codes as given in http://www.utf8-chartable.de/
    2. Length of filename (including path to file): 256 characters or more.
    3. Length of filename extension: “dot” followed by three characters or more
2. Minimum Directory structure:
    1. Number of files or directories in root directory: 256 entries or more.
    2. Number of files or directories per directory (except root): 2048 entries or more
    3. Depth of directory hierarchy: 9 or more (assuming at least 256 entries per directory level).
    4. Directory size: 128 KB or more (assuming 64 bytes per directory entry).
3. File Size: 64 MB or more.
4. Number of files per volume: 41,600 files or more (assuming 650 MB CD with 16 KB files.
5. Support for removable media.
6. Support for bootable/non-bootable volume.

5.2. Operating System

The CDB standard is Operating System (OS) independent; it does not mandate the use of a specific OS. However, compliance to this standard does require that the operating system be able to support a minimal set of capabilities.

Any operating system implementing the CDB standard *should* support at a minimum the following basic OS properties:

- Byte-stream random file access
- 32-bit integers, natively
- A 32-bit address space
- Floating point support (per IEEE-754), natively

- 2GB virtual address space per process
- Memory paging
- Network communication

5.3. System Hardware Independence

Implementation of the CDB standard is hardware independent. The CDB standard does not mandate the use of particular hardware platforms. Furthermore, any general-purpose hardware compatible with modern Operating Systems (OS) can be used for a CDB implementation.

Implementation of the CDB standard assumes that the system hardware shall support, as a minimum, the subsystems described in the following requirement.

Hardware memory for systems implementing the CDB standard *should* support:

- 8-bit, 16-bit, and 32-bit signed and unsigned integers, natively
- A 32-bit address space
- 32-bit and 64-bit double precision floating point values (IEEE-754), natively
- 2 GB virtual address space
- Virtual memory space

Immediate and indirect memory addressing modes

Chapter 6. Informative Implementation Guidance

6.1. Clarification: Publisher Considerations

The CDB Standard does not provide guidelines regarding its implementation within specific vendor SE toolsets and vendor simulation architectures. This clearly falls outside of the scope of the CDB Standard. The CDB Standard is focused a physical and logical storage structure and guidance on storage formats for information (i.e., it merely describes data) for use in simulation.

The CDB model lends itself to a real-time implementation within simulation architectures. This capability requires that the vendor's client-device be adapted with a Run-Time publishing (RTP) software function which transforms the CDB structured data into the client-device's internal legacy/proprietary format. This is a new concept for the simulation industry and consequently there is considerable confusion regarding the implementation of Off-line and Run-Time Publishers (RTPs). While much of the attention has focused on RTPs, a similar set of considerations apply to the implementation of an off-line CDB capability (CDB is used as a Refined Source Data Repository). In this latter case, the capability requires that the vendor develop an off-line CDB import function which ingests the CDB into their Synthetic Environment Creation toolset. Once imported, the vendor toolset could produce the vendor's proprietary data format through an off-line compilation function.

By definition, the function of an RTP is to bridge the "gap" (or adapt) between CDB data schema and the client-device's internal (proprietary) data schema. Since this gap is unknown, it is impossible in this addendum to provide hard-and-fast rules and detailed estimates for the implementation of an RTP (or a CDB import function).

Note that there are many alternatives open to a vendor when considering the level of compliancy he wishes to achieve. The level-of-effort is essentially a function of the level of compliancy the vendor wishes to achieve, and the size of the intrinsic "gap" between the CDB data schema and his device's internal schema.

Nonetheless, this section highlights aspects of the CDB that are particularly significant when considering such implementations. These aspects dominate the level-of-effort required to achieve ideal CDB compliancy.

The CDB Standard limits itself to a description of a data schema for Synthetic Environmental information (i.e. it merely describes data) for use in simulation. The CDB Best Practices provide rigorous guidance of the semantic meaning for each dataset, each attribute and establishes the structure/organization of that data as a schema comprised of a folder hierarchy and files with internal (industry-standard) formats. This ensures that the all CDB data is understood, interpreted and efficiently accessed in the same way by each client-device. The CDB standard does not include detailed guidelines regarding off-line database compliers or runtime publisher implementations, since this would be tantamount to dictating internal vendor formats which are by their very nature proprietary.

The CDB Standard DOES NOT specify:

- The implementation details of an off-line CDB import function that can then be used to compile the imported Synthetic Environmental data into one or more types of proprietary runtime databases (only the client-device vendor has this knowledge and control);
- The implementation details or algorithms of runtime publishers attached to specific client-device (only the client-device vendor has this knowledge and control); or
- The implementation details or algorithms of client-devices that use CDB data (only the client-device vendor has this knowledge and control).

While the CDB standard does not govern the actual implementation of client-devices, it is expected that the CDB standard will have a "unifying" effect on the implementation of each vendor's client-device by virtue of the fact that they will share the exact same Synthetic Environmental data. It is expected that side-by-side comparisons will be easier to undertake due to the fact that devices will run off the exact same runtime data. Prior to the advent of the CDB standard, side-by-side comparisons were considerably more difficult to undertake due to the fact the entire SE creation chain starting from raw source was implicated in such evaluations.

If we set aside legacy considerations, the simplest approach to adopting the CDB would require that client-devices ingest the CDB natively, i.e., client-devices would handle all of the CDB data schema/semantics without any off-line or run-time intermediary.

In practice however, most vendors have extensive legacy SE assets and cannot afford to obsolesce these. As a result, most client-devices must continue to support their own proprietary legacy runtime databases. Given these considerations, two solutions are possible.

1. No change to the Client-device: In this approach, vendors have chosen to achieve an off-line CDB capability (CDB is used as a Refined Source Data Repository). This capability requires that the vendor develops an off-line CDB import function which ingests the CDB into his Synthetic Environment Creation toolset; once imported, the toolset produces (as always) the vendor's proprietary data format through an off-line compilation function.
2. Level-of-Effort of a Publisher Implementation

The following discussion attempts to qualify the level-of-effort to achieve CDB compliancy. The discussion applies equally to both paradigms, i.e., the CDB Runtime Publishing paradigm and the CDB import-then-compile paradigm.

In the case where a client-device already supports most of data schema and semantics concepts of the CDB, then the RTP (or import-then-compile) software is proportionally less complex. For instance, if an IG already supports the concepts of tiles, of levels-of-detail, of layers and understands the concepts of datasets such as terrain texture, gridded terrain elevation, gridded terrain materials, etc. then there is a modest amount of work to be performed by an RTP.

NOTE	The level-of-effort in adopting the CDB model is proportional to the difference between the CDB data schema and client-device's internal proprietary data schema.

Clearly, the algorithmic complexity of an RTP and the computational load imposed on the RTP is directly proportional to the above-mentioned "gap". The larger the "gap", the more expensive a RTP is to develop and the more computational resources need to be allocated to implement it. Conversely, with a smaller "gap", the RTP development is straightforward and relatively few

computational resources need to be allocated to this function.

In order to assess the level-of-effort to adopt the CDB model, the vendor must first evaluate the similarity of data schemas between the CDB and his client-device, in particular, the vendor must assess whether they espouse the following fundamental CDB concepts:

- Independent Tiles (used for paging of all data)
- Independent Levels-of-Detail (for all data)
- Independent Layers (Dataset Layering)
- Following dataset concepts and semantics:
  - Semantic understanding of all CDB layers
  - Geo-gridded data layers consisting of terrain altimetry, terrain texture, terrain materials/mixtures
  - Geo-defined vector features (points, lineals, areals)
    - With/without modeled 3D representations
    - Feature attribution
  - 3D Modeled representation of features (using a data schema similar to or equivalent to OpenFlight)
    - Instanced geotypical models
    - Instanced model geometry
    - Instanced model texture
    - Non-instanced geospecific models
  - Conforming of features to terrain skin (e.g. height conforming)
  - Topological networks
  - JPEG-2K compression
  - Generation of device-specific NVG/FLIR rendering parameters for light-points and materials

In the case where a client-device does not intrinsically support one or more of the above-mentioned CDB concepts, the RTP must perform SE conversions that will likely fall beyond those of mere format/structure manipulations. Such conversions may affect the precision of the data, its semantic meaning, etc. and thus can compromise certain aspects of runtime correlation.

The CDB data model favors modern up-to-date implementations of client-devices. In effect, the level-of-effort to develop an RTP for an obsolete legacy device is likely to be greater than for a modern device. This is because early approaches in digital computer based flight simulation were more severely constrained by severe hardware, software and data source limitations. Consequently, simulation engineers made important compromises between a subsystem's targeted fidelity and its level of generality, scalability, abstraction, and correlation with other simulator client-devices. In many cases, engineers reverted to complex support data structures (generated off-line) in order to reduce the computational load at runtime.

A classic example of this was the use of Binary Separation Planes (BSPs) data structures [3: Such

BSP data structures where required by most IG vendors prior to ~1995 due to the fact that the IGs did not have sub-pixel level Z-buffer capability.] which were required prior to the widespread adoption of Z-buffers by the IG vendors. The CDB standard does not make provisions for this and as such, the RTP for legacy BSP-based IG devices would be burdened with the rather difficult task to generate BSPs in real-time.

Given their tremendous benefit, the concepts of paging (e.g. tiles) and levels-of-details have steadily been adopted by simulation vendors over the past 15-20 years and have been applied to most datasets, notably terrain and imagery datasets. (See Appendices G and F of the Volume 2: OGC CDB Core Model and Physical Structure Annexes for a rationale for Tiles and Levels-of-detail). As a result, it is not expected that the CDB tiles and LOD concepts will be a problem for most vendors. Note however that CDB applies these two concepts to ALL dataset layers including vector features and 3D models.

6.1.1. Client-Devices

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

Since it is the client-devices that initiate access to the CDB conformant data store, they must each be theoretically "aware" of at least the geodetic earth reference model [4: http://onlinelibrary.wiley.com/doi/10.1029/EO062i007p00065/abstract]. Otherwise, the contents and the structure of the data store instance can be completely abstracted from the client-device.

6.1.2. Typical Functions Performed by a Publisher Implementation

The following discussion provides a typical list of software functions that must be developed in order to achieve CDB compliancy. The discussion applies equally to both paradigms, i.e. the CDB Runtime Publishing paradigm and the CDB import-then-compile paradigm.

Virtually all simulation client-devices in existence today natively ingest their own proprietary native runtime formats. In order to ingest CDB structured data directly, vendors must adapt the device's software to natively ingest the currently defined CDB formats [5: The number of specified formats will be expanded in future versions of the CDB standard.] (e.g. TIFF, Shape, OpenFlight, etc.) or alternately, they can insert a runtime publisher function that transforms the CDB data formats into legacy client device's native runtime format. The runtime publishing process is performed when the CDB is paged-in from the CDB storage device.

The runtime publishers are nothing more than well-optimized offline publishers capable of responding to the on-demand compilation of datasets as they are being paged-in by the respective

client devices. The function of a runtime publisher is no different than that of a conventional offline database publisher, i.e., it…

1. transforms the assembled data store so that it satisfies the client-device's internal data structure and format
2. transforms the assembled data store so that it satisfies the client-device's internal naming conventions
3. transforms the assembled data store so that it satisfies the client-device's number precision and number representation
4. transforms the assembled data store into parameters compatible with the client device's internal algorithms (typically light parameters, FLIR/NVG parameters, etc.
5. transforms the assembled data store so that it satisfies the client-device's data fidelity requirements
6. transforms the assembled data store so that it satisfies the client-device's performance and internal memory limitations
7. transforms the assembled data store so that it satisfies the client-device's level of-detail representation requirements.

Ideally, the scope of an RTP should be purely limited to manipulations of data format and data structure and internal naming conventions (items a-g above). Under such circumstances, it is possible to achieve perfect runtime correlation between client-devices.

6.1.3. Publisher Implementation Recommendations

The use of the CDB data schema "as-is" by a client-device achieves all of the benefits stated in sections 1.4 and 1.5 of the CDB Standard, namely:

1. Improved SE generation timeline and deployment
2. Interoperable simulation-ready SE
3. Improved client-device robustness/determinism
4. Increase SE longevity
5. Reduced SE storage infrastructure cost
6. Platform independence and scalability
7. SE scalability and adaptability

In the case where a client-device does not adhere to one or more of the above-mentioned "fundamental CDB concepts", fewer of the CDB benefits will be realizable.

For instance, a client-device incapable of dealing with levels-of-detail will not have the same level SE scalability (a benefit explained in section 1.4.7 of the CDB Standard) as one that fully espouses that concept. While the latter may be acceptable, it is clearly a less-compliant and an inferior implementation of the CDB than the former.

Changes to the modeled representation of features are generally not advisable since it invariably affects the accuracy of the modeled representation. Most image generators in use today can ingest a

(one-for-one correspondence) the CDB modeled polygonal representation of 3D features. However, in the case of terrain, there are two dominant approaches in industry, either a regular grid with LODs or alternately, the Terrain Irregular Network (TIN) mesh. The CDB Standard has opted for the former given its greater scalability, determinism and compatibility with tiling schemes. Clearly, implementations where such conversions are not necessary are advantaged and provide more of the above-mentioned CDB benefits.

Furthermore, the CDB is designed to provide both the semantic (e.g. vector data/attribution) and the modeled representation of features. Since the CDB Standard and associated Best Practices provides both, it is not advisable to ignore or replace the modeled representation (if provided) nor is it advisable to synthesize a non-CDB modeled representation if none was supplied within the CDB. While the CDB Standard does not forbid vendors to interpret CDB feature data for the purpose of procedurally synthesizing more detailed feature data or synthesizing modeled data from the feature data, *this practice is not recommended as this would severely compromise correlation and inter-operability*. In the context of correlated synthetic environments, such approaches are viable if and only if all client-devices in a federation are equipped with the exact same procedural algorithms. Currently, this is not possible because there are no industry-standard, open-source procedural algorithms endorsed by all simulation vendors.

In the case of the CDB Runtime Publishing paradigm and the CDB import-then-compile paradigm, it is not advisable to ignore or replace the modeled representation (if provided) nor is it advisable to synthesize a non-CDB modeled representation if none was supplied within the CDB.

6.2. Use of a CDB conformant data store as an Off-line Repository

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

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

workflow can be allocated by specialty (e.g., altimetry, satellite imagery, vector data) or by geographic footprint.

- Straightforward SE Archival and Recovery

Note that the use of the use of CDB conformant data store as an offline repository does not impose any change to the simulation training equipment (i.e., no modifications to client-devices are required [6: Or alternately, runtime publishers need not be developed for client-devices]). However, the deployment of the synthetic environment is similar to the conventional approaches used in industry requiring the time-consuming, storage-intensive, off-line compilation of proprietary runtime databases to each client-device. Furthermore, the computing demands on the data store generation facility are significantly greater because the entire data store must be published off-line for each client-device before it can be deployed. These costs rapidly escalate with the complexity and size of the synthetic environment, the number of supported client-devices and the number of supported training facilities. For complex data stores, these costs can far outweigh the costs of the runtime publishers attached to each simulator client-device.

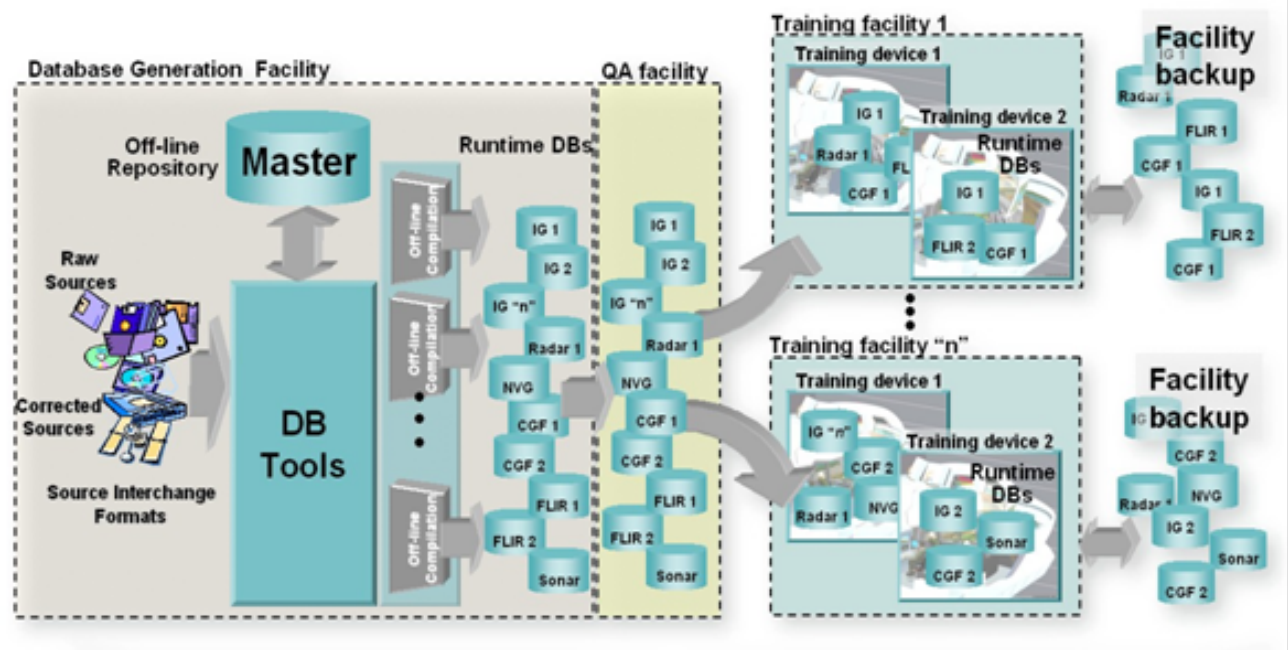


Figure 1-1. Use of CDB Conformant Database as an off-line Database Repository

In most modern SE tool suites in-use today, the Data Preparation step shown in Figure 1-2: SE Workflow with a CDB structured data store as an Off-line Repository consists of many sub-steps usually applied in sequence to each of the datasets (aka layers) of the SE. In effect, this aspect of the modeler's responsibilities is virtually identical to that of a GIS [7: *G*eographic *I*nformation *S*ystems] specialist. As a result, many of the simulation equipment vendors offer SE authoring tools that integrate best-of-breed COTS [8: *C*ommercial-*O*ff-*T*he-*S*helf] GIS tools into their respective tool suites. The steps include the following.

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

automatically, particularly if there is some redundancy across the layers of data.

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

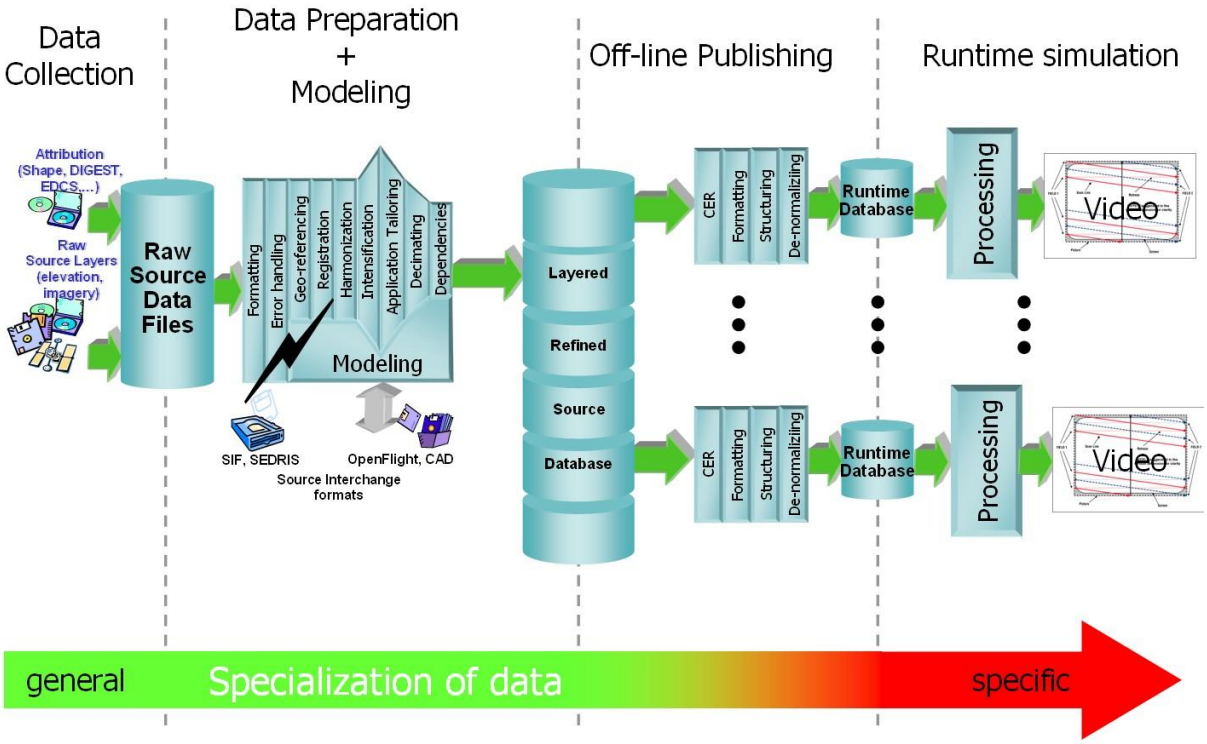


Figure 1-2. SE Workflow with CDB as an off-line Repository

The effort expended during the Data Preparation and Modeling step is mostly independent of the targeted simulation devices and the targeted applications. Consequently, the results of the data preparation step can be stored into a Refined Source Data Store (RSDS) and then re-targeted at modest cost to one or more simulation devices.

The standardization of simulation data stores can greatly enhance their portability and reusability. The CDB Standard and associated OGC Best Practices offers a standardized means to capture the effort expended during the Data Preparation and Modeling step. In effect, a CDB structured database becomes a master repository where refined source can be “accumulated” and managed under configuration control.

While standardization of format/structure is essential to achieve high portability, interoperability and reuse, the SE content must be ideally developed so that its content is truly independent of the

training application. Therefore, we strongly recommend that the SE content of the CDB structured repository be developed to be independent of the training application.

Historically, SEs were developed for a single, targeted simulation application (e.g., tactical fighter, civil and air transport, rotary wing, or ground/urban warfare). In effect, the intended training application played an important role in determining the RSDB content because SE developers were constrained by the capabilities of the authoring tools and of the targeted simulation device. Unfortunately, this tailoring of SE was performed too early during the SE workflow and severely limited the applicability and re-use of the SE. Application tailoring can require either data intensification [9: Data Intensification is the process of augmenting or deriving added detail from the information found in the raw data. For instance, intensification can be used to augment flattened terrain imagery with 3D cultural detail relief. A typical example of this consisting in populating forested areas found in the terrain imagery with individual three-dimensional trees.] or data decimation [10: Data Decimation is the process of removing or simplifying the informational content found in the raw data. For instance, decimation can be used to transform individually modeled buildings into simplified city blocks or to reduce the resolution of terrain imagery. Data decimation is usually undertaken to ensure that the SE falls within the capabilities of the targeted simulator system.] .

Once the SE developer has completed his work in creating the various data layers of the RFDS, he must offline publish (aka “compile”) the SE into one or more device-specific data publishing steps. As we will discuss in section 6.4, Use of CDB structured data store as a Combined Off-line and run-time data store Repository, the device-specific off-line compilation step can be entirely omitted if the targeted training equipment is CDB-compliant.

While an off-line publishing approach does not offer all of the benefits described in this section, it nonetheless provides an easy, low-effort, migration path to CDB. Any equipment vendor can easily publish the data into their proprietary runtime format. Firstly, the publishing process is facilitated by the fact that the CDB standard provides guidance on how to use industry standard formats. However, the CDB model goes much further in that it specifies how to use these formats in a global, standardized data model suited to high-end real-time simulations. This greatly facilitates the work of SE developers. Thus, the CDB model provides a far simpler and straightforward means of interchanging refined source data.

6.3. Use of a CDB conformant data store as a Combined Off-line and Run-time Data store Repository

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

In addition to the benefits outlined in section 6.3, the use of the CDB conformant data store as a combined off-line and run-time repository offers many additional benefits.

- SE Plug-and-Play Portability and Interoperability across CDB-compliant simulators and simulator confederacies (be it tactical air, rotary, urban/ground, sea).

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

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

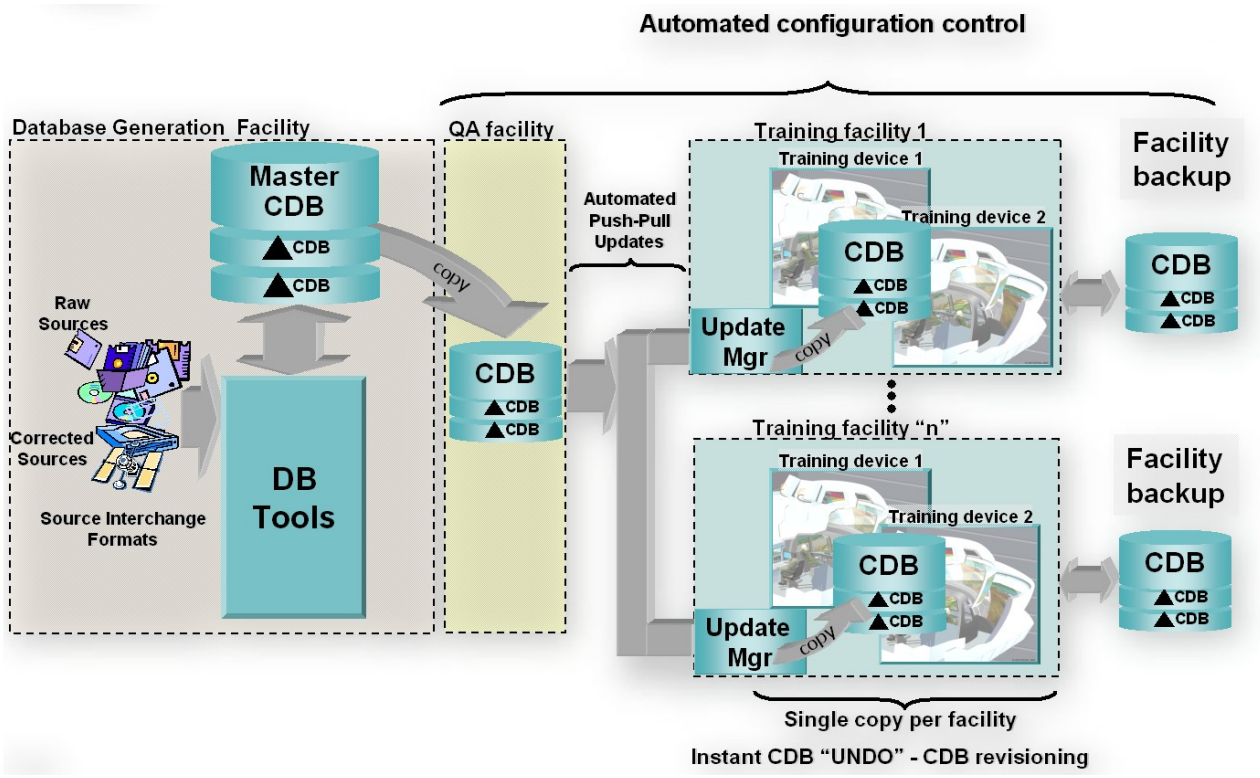


Figure 1-3. Use of CDB as an Off-line and On-line Data Store Repository

The CDB standard associated Best Practices specify formats and conventions related to synthetic environments for use in simulation. However, many additional benefits can be garnered if a CDB structured data store is also used as an online data store repository. This is particularly true when one considers the effort expended in the deployment of the synthetic environment to the training and/or mission rehearsal facilities.

When used as an online data store repository, there is no need to store and maintain off-line published versions of the data store for each client-device (as illustrated in Figure 1-3). As a result,

the storage and computing demands on the data store generation facility are significantly lowered. This is especially true of data store generation facilities whose mandate involves the generation of complex synthetic environments for use by several training facilities.

Figure 1-4: SE Workflow with CDB as Combined Off-line/Runtime Data Store Repository, illustrates the simplified database generation workflow resulting from a data store that is used as both an offline and a runtime SE repository.

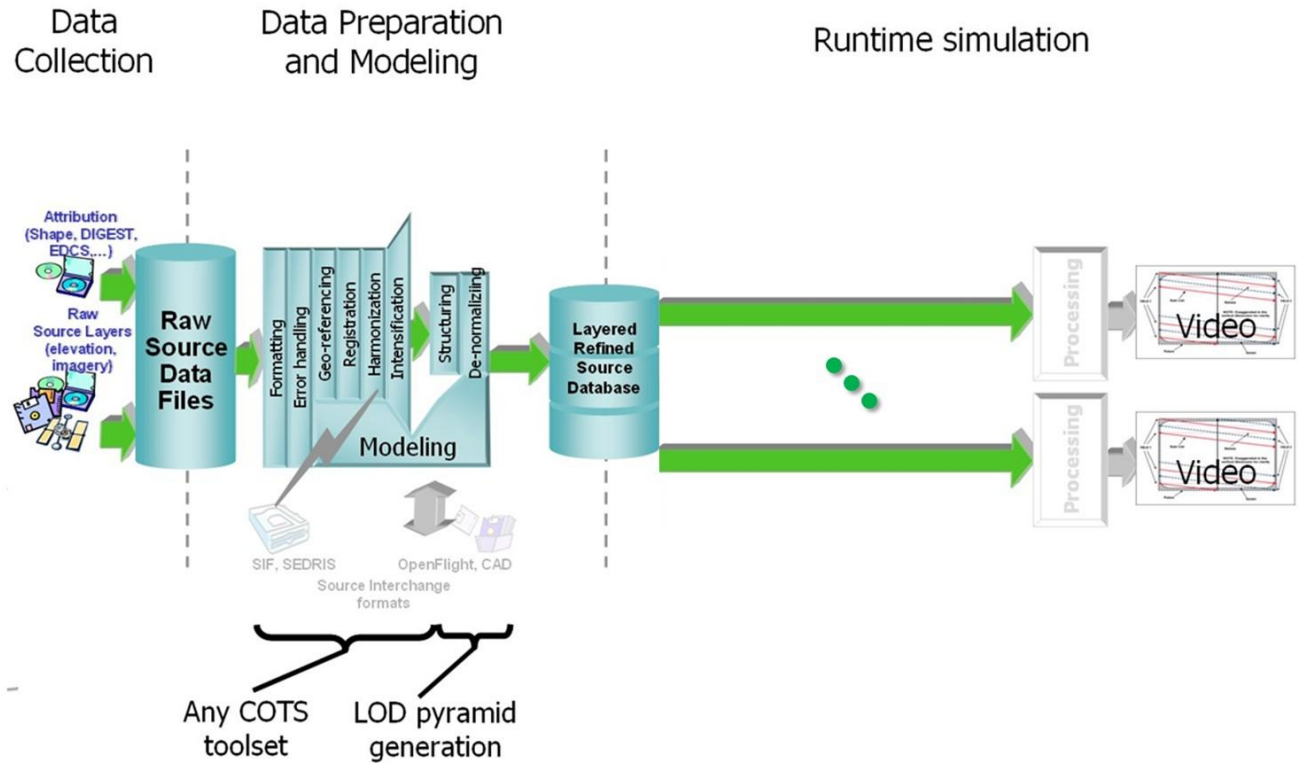


Figure 1-4. SE Workflow with CDB as Combined Off-line/Runtime Data Store Repository

This approach permits the CDB representation of the synthetic environment to be "dissociated" from the resolution, fidelity, precision, structure and format imposed by the internals of client-devices. Compliancy to the CDB standard can be achieved either by modification of the client-device internal software to make it CDB-native or by inserting a runtime publishing process that transforms the CDB structured data into the client-device's legacy native runtime format. In the later case, this process is done in real-time, on a demand-basis, as the simulator "flies" within the synthetic environment. Note that since the simulated own ship [11: Own ship is the object you are on. Target ship is the object you are watching.] moves at speeds that are bounded by the capabilities of the simulated vehicle, it is not necessary to instantly publish the entire synthetic environment before undertaking a training exercise; the runtime publishers need only respond to the demands of the client-devices. When the simulated own-ship's position is static, runtime publishers go idle. As the own ship starts advancing, client-devices start demanding for new regions, and runtime publishers resume the publishing process. Publishing workload peaks at high-speed over highly resolved areas of the synthetic environment.

Note that virtually all simulation client-devices in existence today natively ingest proprietary native runtime formats. As a result, a runtime publisher is required to transform the CDB structured data into legacy client device's native runtime format. The runtime publishing process is performed when the CDB conformant database is paged-in from the CDB storage device. Volume 7, OGC CDB

Data Model Guidance provides a set of guidelines regarding the implementation of Runtime Publishers.

6.4. Example Implementation of a CDB Structured Data Store on a Simulator [1: Legacy simulator client-devices can be readily retrofitted for compatibility with the CDB Standard by inserting a runtime publisher in their SE paging pipeline.]

This section illustrates a possible implementation architecture of the CDB Standard on a flight simulator. The standard does not mandate particular simulator architecture or the use of specific computer platforms. The selected implementation varies with the required level of fidelity and performance of the simulator and its client-devices.

As shown in Figure 1-5: *Typical CDB Implementation on a Suite of Simulators*, a typical implementation of a CDB compliant system consists of the following main components.

1. Data Store Generation Facility (DBGF) and CDB Master Store: A geographically co-located group of workstation(s), computer platforms, input devices (digitizing tablets, etc.), output devices (stereo viewers, etc.), modeling software, visualization software, database server, off-line publishing software and any other associated software and hardware used for the development/modification of the data store. The CDB Master Store consists of a mass storage system (typically a storage array) and its associated network. It is connected to a dedicated DBGF Server.
2. Update Manager (UM): The Update Manager software consists of both client and server software. The Update Manager Server (UMS) software is located at the DBGF. It manages the data store updates (versions) and runs in the same platform as the DBGF Server. The Update Manager Client (UMC) software is located at the Simulator Facility and runs on the Update Manager Platform shown in Figure 1-5: *Typical CDB Implementation on a Suite of Simulators*. The UMC communicates with the UMS to transfer the data store (partial or complete copy) and its updates.
3. Simulator Facility CDB Data Store Repository: The simulator repository consists of a mass storage system (typically a storage array) and its associated network infrastructure. It is connected to the UMC (primarily for update purposes) and the servers (for simulator client-device runtime access).
4. CDB servers: An optional [12: Optionally needed for a large-scale CDB repository whose storage system is based on a Storage Area Network (SAN).] gateway to mass storage and applicable infrastructure. The CDB servers access, filter and distribute data in response to requests from the simulator runtime publishers.
5. Runtime publishers: A term used to describe the computer platforms, and the software that translates and optimizes, at runtime, CDB synthetic environment data store to a client-device specific legacy runtime format. Data is pulled from the CDB server and in turn published in response to requests from its attached simulator client-device.
6. Simulator client-devices: Are simulation subsystems (IGs, radar, weather server, Computer

Generated Forces (CGF) terrain server, etc.) that require a complete or partial synthetic representation of the world. CDB runtime clients may require a CDB runtime publisher to convert the CDB into a form they can directly input.

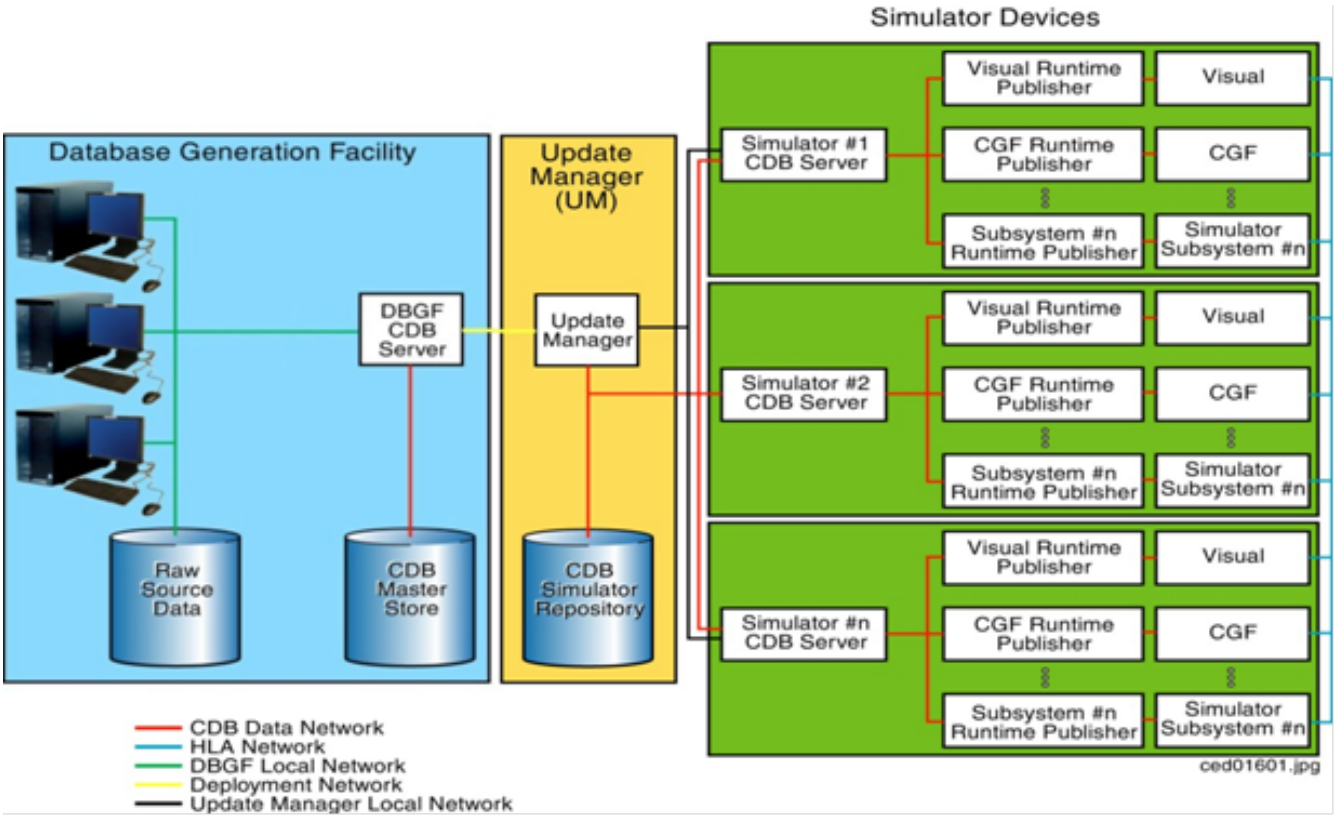


Figure 1-5. Typical CDB Implementation on a Suite of Simulators

6.4.1. Data Store Generation Facility (DBGF)

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

6.4.2. Database Generation Flow

The CDB Model considerably simplifies the data store generation process, particularly all aspects of data store generation that deal with data store layering, formatting, structure and level-of-detail.

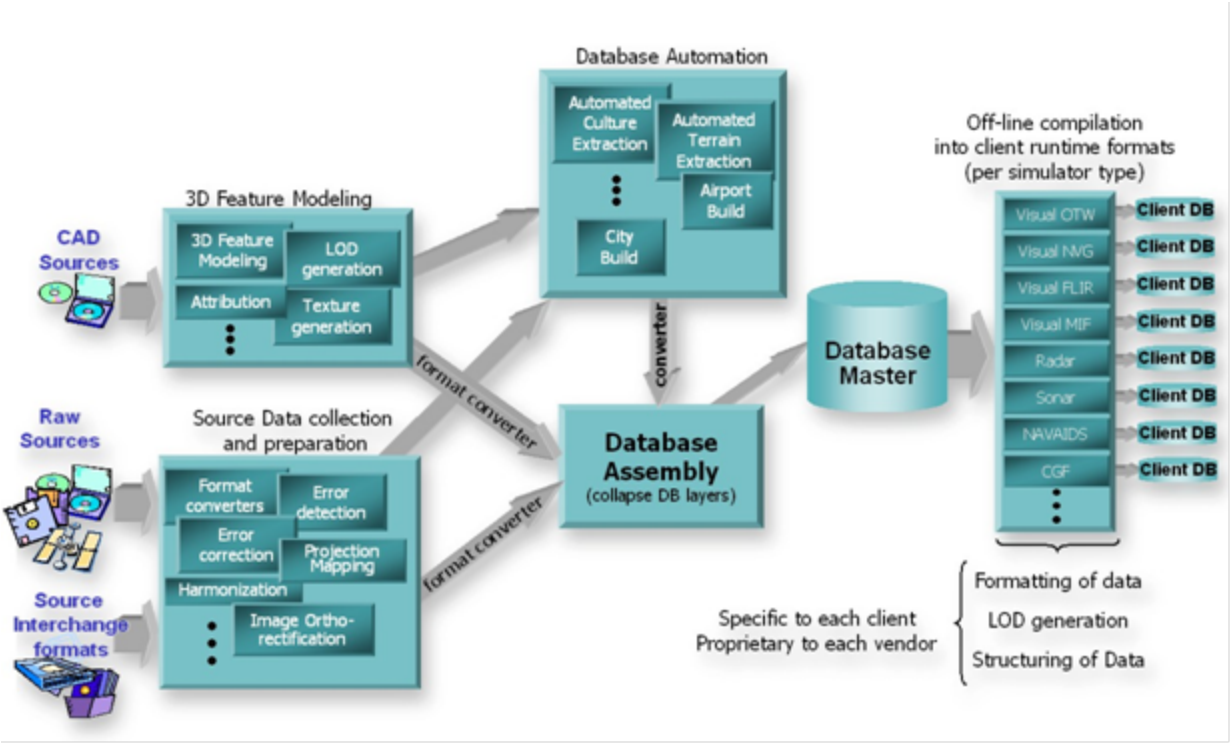


Figure 1-6. Typical DB Generation - CDB Used as a DB Repository

Figure 1-6: *Typical DB Generation - CDB Used as DB Repository* and Figure 1-7: *Typical DB Generation Flow - CDB Used as DB & Sim Repository* illustrate a typical database generation workflow with the database used as a DB workstation repository and the database used as a Repository for the DB workstation and the simulator. Both approaches share the same steps, namely…

1. *Source data collection and preparation:* This step usually involves the loading of raw (usually) uncorrected data and the conversion to formats native to the data store toolset.
2. *Source data preparation:* This step usually involves the detection/correction of errors, the harmonization of the data and the correction of errors. In this context, errors signify all instances where the data fails to meet prescribed criteria. For instance, errors can be as straightforward as corrupted digital data. More subtle forms of errors could be textures that fail to meet various brightness, contrast, chrominance, and distortion criteria. Harmonizing data requires that data sources be coherent with each other. An example of non-harmonized dataset is a terrain imagery mosaic built from pictures taken in different seasons, with different illumination conditions, with/without clouds, etc.
3. *3D modeling of features:* This step involves the creation of 3D representations for culture features (buildings, trees, vehicles, etc.), the creation and mapping of texture patterns/imagery to the geometrical representation, the generation of the model LOD, and the generation of appropriate attribution data so that the simulator can control the model and have it respond to the simulated environment.
4. *Data Store automation:* Modern data processing and validation tools offer an increasing level-of-automation to the modelers, thereby improving the DB generation timeline (for example, a forest tool that controls the placement of individual trees correlated to the underlying terrain imagery). Over the past few years, tool vendors have introduced a broad set of tools aimed at eliminating highly repetitive modeling tasks. This includes tools for runway generation (including the positioning of stripes, lights, signs, markings, etc.), road/railroad generation,

cultural feature extraction from stereo pairs, cultural feature footprint extraction from image classification processes, terrain grid generation from stereo pairs, terrain surface material classification, etc.

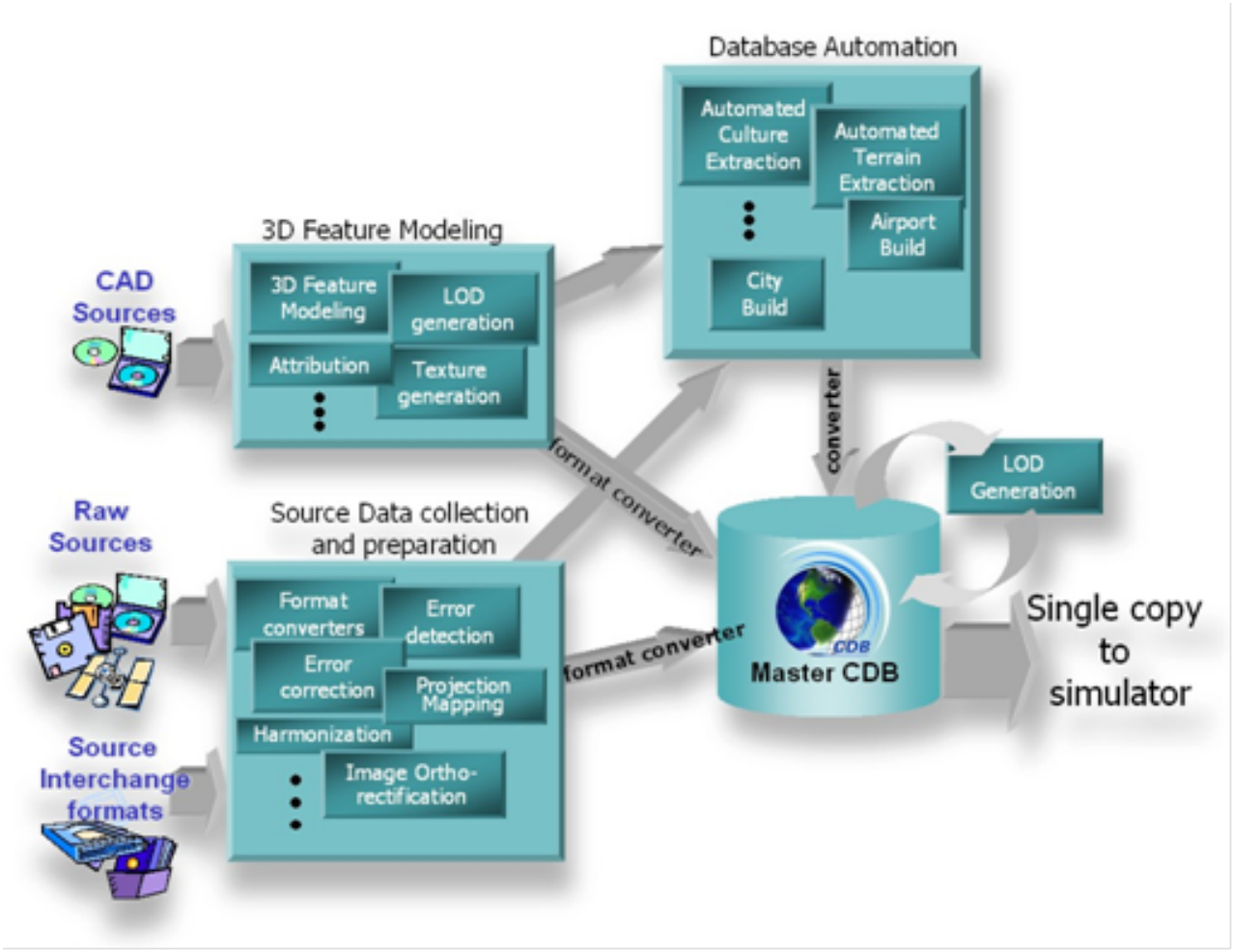


Figure 1-7. Typical DB Generation Flow - CDB Used as DB and Sim Repository

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

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

6.4.3. Update Manager

The creation of the CDB structured data store and subsequent updates are performed at the DBGF. The Update Manager (UM) keeps track of these updates and synchronizes the Simulator CDB Repository to the DBGF. The CDB Standard permits flexible and efficient access of the data store and does so with different levels of granularity. Thus, it is possible to perform modifications to the database on a complete tile, or on individual datasets of a tile. This permits rapid deployment of the data store, a feature that is particularly valuable for mission planning and rehearsal. With few exceptions [13: The only exceptions to this CDB principle are the MinElevation, MaxElevation

datasets which are slaved to the Terrain Elevation dataset and the MaxCulture dataset which is slaved to the GSFeature/GTFeature dataset.], there is no interdependency between datasets and it is possible to modify a dataset (such as the terrain imagery) without reprocessing the complete tile; only the modified dataset requires re-processing. The CDB Standard supports the concurrent creation/modification of the data store with its deployment. Once a tile, a feature set, or a dataset has been processed, it may be transferred to the simulator facility concurrently with other work performed at the DBGF.

Updates to the simulator CDB structured repository are performed by the UM. The simulator CDB repository is configured to provide storage for a (partial or complete) copy of the Data Store Generation Facility (DBGF) master store. The Update Manager transfers the data store and its updates by area of interest, allowing for partial updates or even complete copies of the database. The Update Manager (UM) simulator CDB structured repository is used by one or more co-located simulators to retrieve the data store in real-time.

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

6.4.4. CDB Servers

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

6.4.5. Runtime Publishers

When the CDB structured data store is used as an on-line (or runtime) repository, a set of runtime publishers are required in order to transform the CDB data into legacy client-devices (simulator subsystems) internal format [14: Alternately, client-devices can be designed / modified to natively handle the CDB's data model, thereby obviating the need for a separate runtime publishing step.]. The runtime publishers provide a key role in further enhancing overall algorithmic correlation within and across simulators. Each publisher communicates to the CDB data store server complex and the attached simulator client-device as follows.

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

6.4.6. Simulator Client-devices

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

Visual Subsystems

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

Out-The-Window Image Generator (OTW IG)

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

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

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

Infrared IG

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

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

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

Night Vision Goggles Image Generation

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

Ownship-Centric Mission Functions

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

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

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

Computer Generated Forces (CGF)

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

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

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

Weather Simulation

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

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

Radar

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

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

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

Navigation System

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

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

1. Tactical Air Navigation (TACAN)
2. Automatic Direction Finder (ADF)
3. VHF Omni Range (VOR)
4. Instrument Landing System (ILS)
5. Microwave Landing System (MLS)
6. Doppler Navigation System (DNS)
7. Global Positioning System (GPS)
8. Inertial Navigation Unit (INU)
9. Non-Directional Beacons (NDB)

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

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

NAV uses time invariant information held in CDB such as:

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

6.4.7. CDB Data Store and Model naming Guidance

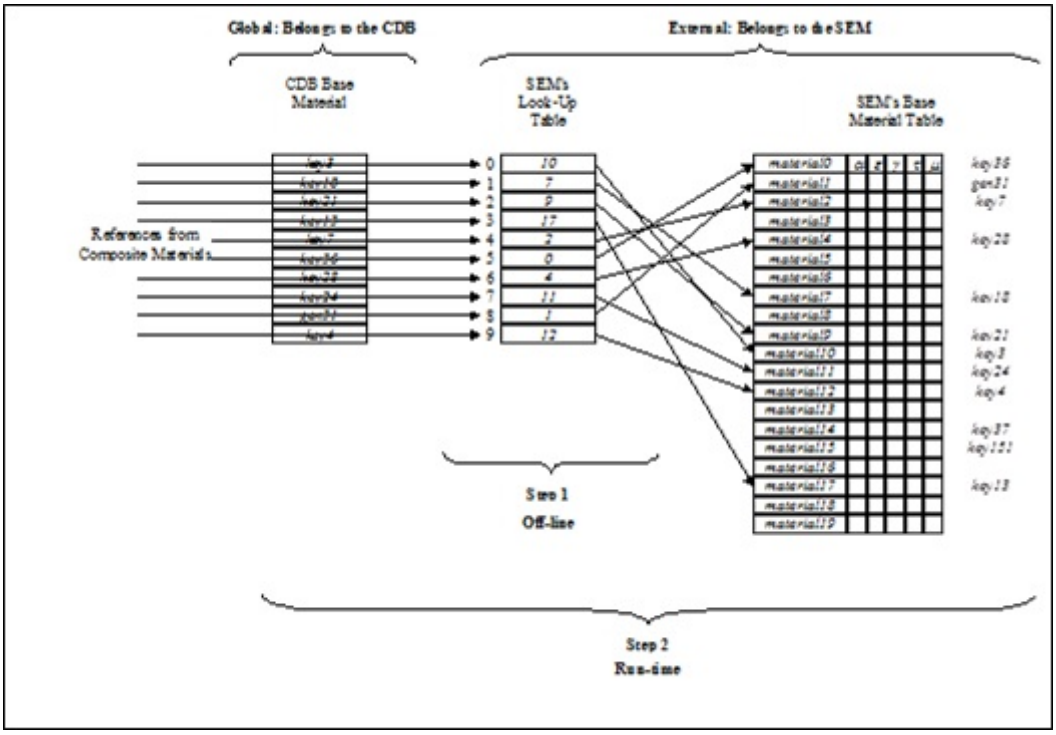
Sensor Simulation and Base Materials linkage

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

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

1. The first step consists in establishing a correspondence between all of the Base Materials in the CDB data store and the Base Materials directly supported by the SEM of the client-device. This is a manual task performed by the SEM specialist(s). The specialist must ensure that his SEM has a corresponding Base Material for each of the CDB Base Materials. In cases where the SEM is simple, it is possible for two or more CDB Base Materials to point to the same SEM Base Material. Alternately the SEM specialist may choose to create new SEM Base Materials that correspond more closely to the CDB's Base Materials. The result of this process is a SEM look-up.

2. The second step is typically undertaken during the CDB data store initialization by the client-device running the SEM. During this initialization phase, the SEM reads the content of the global Base Material Table and the SEM look-up provided by the SEM specialist. This look-up establishes an indirect link between the Base Materials in the CDB data store and the material properties of the client-device's SEM Base Materials. In fact, the indirect link (i.e., the look-up table) can be eliminated if the client device internally builds a Materials Properties Table that uses the CDB material keys directly (as illustrated in Figure 2 11: SEM Base Material Properties Table).



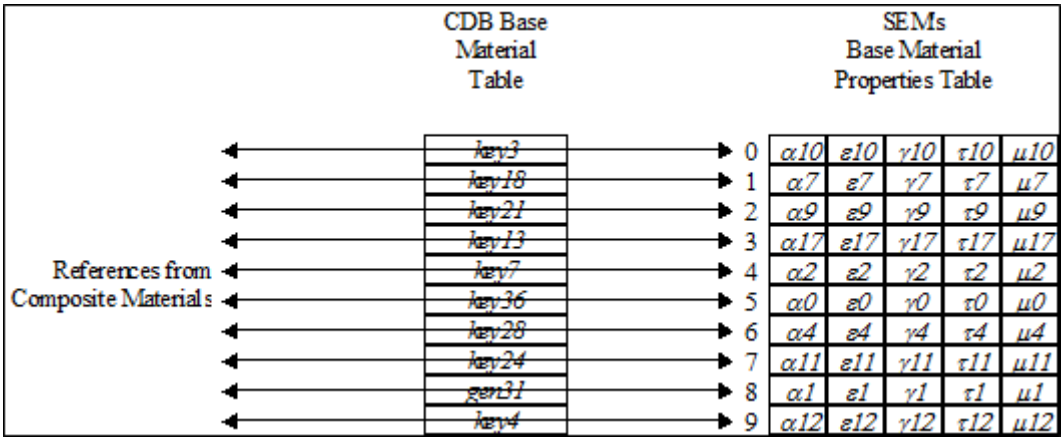


Figure 1-8. SEM Base Material Properties Table

6.4.8. SEM – Materials example

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

water (key3 = "BM_WATER-FRESH", BMT's index 0)

vegetation (key21 = " BM_LAND-LOW_MEADOW", BMT's index 2)

soil (key7 = " BM_SOIL ", BMT's index 4)

sand (key4 = " BM_SAND ", BMT's index 9)

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

key3 to material 8 ("Lake", SEM list's index 8)

key21 to material 3 ("Uncultivated Land", SEM list's index 3)

key7 to material 7 ("Soil", SEM list's index 7)

key4 to material 12 ("Sand", SEM list's index 12)

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

BMT's index 0 is associated to SEM list's index 8

BMT's index 2 is associated to SEM list's index 3

BMT's index 4 is associated to SEM list's index 7

BMT's index 9 is associated to SEM list's index 12

Geospecific viz Geotypical guidance

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

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

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

1. *CDB Storage Size:* The size of the CDB is smaller when the cultural features reference geotypical models rather than geospecific models. This is due to the fact that the modeled representation of geotypical model is not duplicated within each tile – instead, the model appears once in the GTModel library dataset directory. Clearly, a geotypical model is the preferred choice if the modeler wishes to assign and re-use the same modeled representation for a given feature type.

2. *Client-device Memory Footprint:* By assigning a geotypical model to a feature, the modeler provides a valuable "clue" to the client-device that the feature will be instanced throughout the CDB with the same modeled representation. As a result, client-device should dedicate physical memory for the storage of the geotypical models for later use.
3. *GTModel Library Management:* The CDB's Feature Data Dictionary (FDD) is based on the DIGEST, DGIWG, SEDRIS and UHRB geomatics standards. These standards are commonly used for the attribution of source vector data in a broad range of simulation applications. The CDB Feature Data Dictionary acts much like what an English dictionary is to a collection of novels. As a result, it is possible to develop a universal GTModel Library which is totally independent of the CDB content (just like a dictionary is independent of books). This universal GTModel Library can be simply copied into the \CDB\GTModel directory. The structure of the GTModel Library is organized in accordance to the CDB's FDD – in other words, the models are indexed using the CDB Feature Code. The indexing approach greatly simplifies the management of the model library since every model has a pre-established location in the library.
4. *CDB Generation and Update:* As mentioned earlier, the size of the CDB is smaller when the cultural features reference geotypical models rather than geospecific models. This is due to the fact that the modeled representation of geotypical model is not duplicated within each tile – instead, the model appears once in the GTModel library dataset directory. This reduces the amount of time required by the tools to generate and store the CDB onto the disk storage system. The second benefit of geotypical models comes in the case where a modeler wishes to change the modeled representation of one or more geotypical features type across the entire CDB. Changes to the modeled representation of a feature type can easily be performed by simply overwriting the desired model in model library. From then on, all features of that type now reference the updated model – no other changes to the CBD are required.

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

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

the CDB storage system.

6.5. Primer: Line-of-Sight (LOS) Algorithms Using MinElevation and MaxElevation Components

Note: Was A.13 in Volume 2 in original submission

The purpose of the MinElevation and MaxElevation components is to provide the CDB data store with the necessary data and structure to achieve the required level of determinism in the computation line-of-sight calculations with the terrain. The values of each component are with respect to mean sea level. Since both the MinElevation and the MaxElevation values are specified in this standard, any line-of-sight algorithm can rapidly assess an intersection status of the line-of-sight vector with the terrain.

There are three cases to consider:

CASE 1 – No intersection: If all of the LOS Bounding Boxes are above the MinMax Bounding Boxes, then there is no intersection between the line-of-sight vector and the terrain. No further testing is required. (Refer to Figure A-16: Case 1 – No Intersection.)

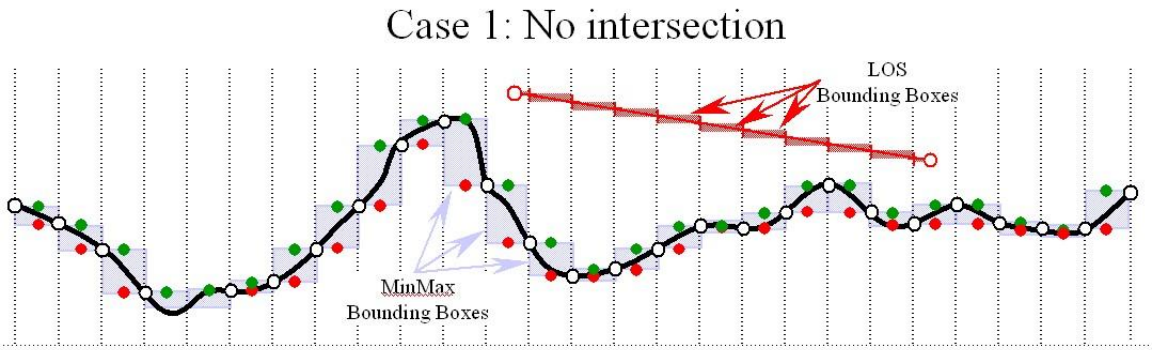


Figure A-16: Case 1 – No Intersection

CASE 2 – Potential intersection: If one or more of the LOS Bounding Boxes overlap with a MinMax Bounding Box, then there is a potential intersection between the line-of-sight vector and the terrain. This step must be repeated with progressively finer level-of-detail versions of the MinElevation and MaxElevation values until Case 1 or Case 3 is encountered. If the finest level-of-detail is reached and the LOS result still yields a potential intersection status (Case 2), then the LOS algorithm must perform a LOS intersection with the finest LOD of the Primary Terrain Elevation component using the prescribed CDB meshing convention. (Refer to Figure A-17: Case 2 – Potential Intersection.)

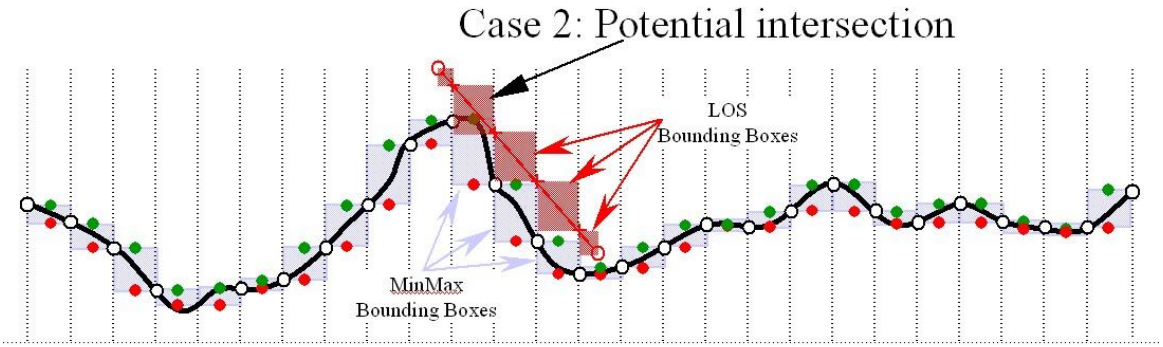


Figure A-17: Case 2 – Potential Intersection

CASE 3 – Intersection: If one or more of the LOS Bounding Boxes are below the MinMax Bounding Boxes, then there is an intersection between the line-of-sight vector and the terrain. No further testing is required to determine whether there is intersection or not. (Refer to Figure A-18: Case 3 – Guaranteed Intersection.) However, to determine the intersection point, the LOS algorithm must perform the following additional steps. If (starting with the LOS point-of-origin) one or more of the LOS Bounding Boxes overlap with a MinMax Bounding Boxes, then there is a potential intersection between the line-of-sight vector and the terrain for that MinMax Bounding Box. This step must be repeated with progressively finer level-of-detail versions of the MinElevation and MaxElevation values until Case 1 or Case 3 is encountered. If the finest level-of-detail is reached and the LOS result still yields a potential intersection status (Case 2), then the LOS algorithm must perform a LOS intersection with the finest LOD of the Primary Terrain Elevation component using the prescribed CDB meshing convention.

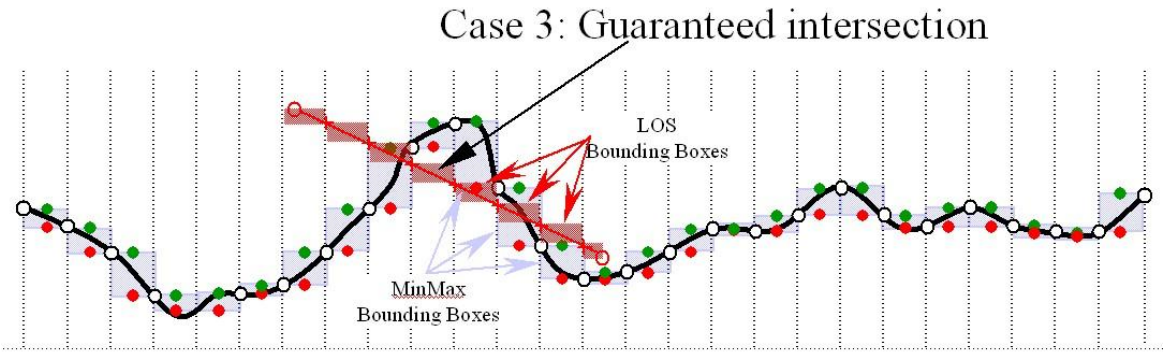


Figure A-18: Case 3 – Guaranteed Intersection

6.6. Gamma Tutorial (Was Annex G, Volume 2)

6.6.1. Introduction

There is nominally no gamma correction done to the stored samples of CDB imagery files. As a result, a gamma of 1/2.2 should be applied to imagery data when viewing it through a (sRGB-calibrated) monitor with gamma of 2.2. The CDB Standard recommends the sRGB IEC 61966-2 standard when performing the calibration of displays (at DBGF or a simulator). The sRGB standard provides the necessary guidelines for the handling of gamma, and of color (in a device-independent fashion) under specified viewing conditions.

It would be convenient for graphics programmers if all of the components of an imaging system were linear. The voltage coming from an electronic camera would be directly proportional to the intensity (power) of light in the scene; the light emitted by a CRT would be directly proportional to its input voltage, and so on. However, real-world devices do not behave in this way.

Real imaging systems will have several components, and more than one of these can be nonlinear. If all of the components have transfer characteristics that are power functions, then the transfer function of the entire system is also a power function. The exponent (gamma) of the whole system's transfer function is just the product of all of the individual exponents (gammas) of the separate stages in the system. Also, stages that are linear pose no problem, since a power function with an exponent of 1.0 is really a linear function. So a linear transfer function is just a special case of a power function, with a gamma of 1.0. Thus, as long as our imaging system contains only stages with linear and power-law transfer functions, we can meaningfully talk about the gamma of the entire system. This is indeed the case with most real imaging systems.

If the overall gamma of an imaging system is 1.0, its output is linearly proportional to its input. This means that the ratio between the intensities of any two areas in the reproduced image will be the same as it was in the original scene. It might seem that this should always be the goal of an imaging system: to accurately reproduce the tones of the original scene. Alas, that is not the case.

When the reproduced image is to be viewed in "bright surround" conditions, where other white objects nearby in the room have about the same brightness as white in the image, then an overall gamma of 1.0 does indeed give real-looking reproduction of a natural scene. Photographic prints viewed under room light and computer displays in bright room light are typical "bright surround" viewing conditions.

However, sometimes images are intended to be viewed in "dark surround" conditions, where the room is substantially black except for the image. This is typical of the way movies and slides (transparencies) are viewed by projection. Under these circumstances, an accurate reproduction of the original scene results in an image that human viewers judge as "flat" and lacking in contrast. It turns out that the projected image needs to have a gamma of about 1.5 relative to the original scene for viewers to judge it "natural". Thus, slide film is designed to have a gamma of about 1.5, not 1.0.

There is also an intermediate condition called "dim surround", where the rest of the room is still visible to the viewer, but is noticeably darker than the reproduced image itself. This is typical of television viewing, at least in the evening, as well as subdued-light computer work areas. In dim surround conditions, the reproduced image needs to have a gamma of about 1.25 relative to the original scene in order to look natural.

The requirement for boosted contrast (gamma) in dark surround conditions is due to the way the human visual system works, and applies equally well to computer monitors. Thus, a modeler trying to achieve the maximum realism for the images it displays really needs to know what the room lighting conditions are, and adjust the gamma of the displayed image accordingly.

If asking the user about room lighting conditions is inappropriate or too difficult, it is reasonable to assume that the overall gamma (viewing_gamma as defined below) is somewhere between 1.0 and 1.25. That's all that most systems that implement gamma correction do.

According to PNG (Portable Network Graphics) Specification Version 1.0, W3C Recommendation 01-

October-1996 Appendix, Gamma Tutorial,

(*http://www.w3.org/TR/PNG-GammaAppendix*):

“All display systems, almost all photographic film, and many electronic cameras have nonlinear signal-to-light-intensity or intensity-to-signal characteristics. Fortunately, all of these nonlinear devices have a transfer function that is approximated fairly well by a single type of mathematical function: a power function. This power function has the general equation

output = input ^ gamma

where ^ denotes exponentiation, and “gamma” (often printed using the Greek letter gamma, thus the name) is simply the exponent of the power function.

By convention, “input” and “output” are both scaled to the range [0..1], with 0 representing black and 1 representing maximum white. Normalized in this way, the power function is completely described by a single number, the exponent “gamma.”

So, given a particular device, we can measure its output as a function of its input, fit a power function to this measured transfer function, extract the exponent, and call it gamma. We often say “this device has a gamma of 2.5” as a shorthand for “this device has a power-law response with an exponent of 2.5”. We can also talk about the gamma of a mathematical transform, or of a lookup table in a frame buffer, so long as the input and output of the thing are related by the power-law expression above.

Real imaging systems will have several components, and more than one of these can be nonlinear. If all of the components have transfer characteristics that are power functions, then the transfer function of the entire system is also a power function. The exponent (gamma) of the whole system’s transfer function is just the product of all of the individual exponents (gammas) of the separate stages in the system.

Also, stages that are linear pose no problem, since a power function with an exponent of 1.0 is really a linear function. So a linear transfer function is just a special case of a power function, with a gamma of 1.0.

Thus, as long as our imaging system contains only stages with linear and power-law transfer functions, we can meaningfully talk about the gamma of the entire system. This is indeed the case with most real imaging systems.”

In an ideal world, sample values would be stored in floating point, there would be lots of precision, and it wouldn’t really matter much. But in reality, we’re always trying to store images in as few bits as we can.

If we decide to use samples that are linearly proportional to intensity, and do the gamma correction in the frame buffer LUT, it turns out that we need to use at least 12-16 bits for each of red, green, and blue to have enough precision in intensity. With any less than that, we will sometimes see “contour bands” or “Mach bands” in the darker areas of the image, where two adjacent sample values are still far enough apart in intensity for the difference to be visible.

However, through an interesting coincidence, the human eye’s subjective perception of brightness

is related to the physical stimulation of light intensity in a manner that is very much like the power function used for gamma correction. If we apply gamma correction to measured (or calculated) light intensity before quantizing to an integer for storage in a frame buffer, we can get away with using many fewer bits to store the image. In fact, 8 bits per color is almost always sufficient to avoid contouring artifacts. This is because, since gamma correction is so closely related to human perception, we are assigning our 256 available sample codes to intensity values in a manner that approximates how visible those intensity changes are to the eye. Compared to a linear-sample image, we allocate fewer sample values to brighter parts of the tonal range and more sample values to the darker portions of the tonal range.

Thus, for the same apparent image quality, images using gamma-encoded sample values need only about two-thirds as many bits of storage as images using linear samples.

If we consider a pipeline that involves capturing (or calculating) an image, storing it in an image file, reading the file, and displaying the image on some sort of display screen, there are at least 5 places in the pipeline that could have nonlinear transfer functions. Let's give each a specific name for their characteristic gamma:

1. Camera_gamma (γ_c):The characteristic of the image sensor.
2. Encoding_gamma (γ_e): The gamma of any transformation performed by the software writing the image file.
3. Decoding_gamma (γ_d): The gamma of any transformation performed by any software reading the image file.
4. LUT_gamma (γ_{lut}): The gamma of the frame buffer LUT, if present.

In addition, let's add a few other names:

1. File_gamma (γ_f): The gamma of the image in the file, relative to the original scene, i.e.

   $\gamma_f = \gamma_c \gamma_e$

2. DS_gamma (γ_{DS}): The gamma of the "display system" downstream of the frame buffer. In this context, the term display system encompasses everything after the frame buffer, that is

   $\gamma_{DS} = \gamma_{lut} \gamma_{crt}$

3. Viewing_gamma (γ_v): The overall gamma that we want to obtain to produce pleasing images generally 1.0 to 1.25.

When the file_gamma is not 1.0, we know that some form of gamma correction has been done on the sample values in the file, and we call them "gamma corrected" samples. However, since there can be so many different values of gamma in the image display chain, and some of them are not known at the time the image is written, the samples are not really being "corrected" for a specific display condition. We are really using a power function in the process of encoding an intensity range into a small integer field, and so it is more correct to say "gamma encoded" samples instead of "gamma corrected" samples. The CDB standard does not rely on such gamma encoding in order to achieve smaller integer number representations. Instead, the CDB standard relies on standard compression algorithms to achieve an efficient representation of color imagery. [15: The JPEG-2000 standard is based on the *sRGB* default color space per the IEC 61966-2-1 Standard which calls for a

gamma 2.2 under the specified viewing conditions]

When displaying an image file on the simulator, the image-decoding software is responsible for making the overall gamma of the system equal to the desired viewing_gamma, by selecting the decoding_gamma appropriately. If the viewing condition is different from the specification, then the decoding process must compensate. This can be done by modifying the gamma values in equation G-1 below by the appropriate factor. If one does modify the gamma values in equation G-1 below, extreme care must be taken to avoid quantization errors when working with 24 bit images. The display_gamma should be measured (and known) for the display rendering the image (either at the DB generation workstation or the simulator). The correct viewing_gamma depends on lighting conditions, and that will generally have to come from the user. In dimly lit office environments, the generally preferred value for viewing gamma is in the vicinity of 1.125 [16: Historically, viewing gammas of 1.5 have been used for viewing projected slides in a dark room and viewing gammas of 1.25 have been used for viewing monitors in a very dim room. This very dim room value of 1.25 has been used extensively in television systems and assumes a ambient luminance level of approximately 15 lux (or 1.4 ft-lb). The current proposal assumes an encoding ambient luminance level of 64 lux (or 5. ft-lb) which is more representative of a dim room in viewing computer generated imagery or a FAA level-D approved flight simulator visual system. Such a system assumes a viewing gamma of 1.125 and is thus consistent with the ITU-R BT.709 standard.]. In many digital video systems, camera_gamma is about 0.5. CRT_gamma is typically 2.2, while encoding_gamma, decoding_gamma, and LUT_gamma are all 1.0. As a result, viewing_gamma ends up being about 1.125. Coincidently, this happens to be the optimal viewing gamma for an ambient luminance level of 64 lux or 5 ft-lbt.

$$\gamma_c \gamma_d \gamma_{DS} = \gamma_v$$ $$\gamma_c \gamma_d \gamma_{lut} \gamma_{crt} = \gamma_v$$ $$0.511 * 1.0 * 1.0 * 2.2 = 1.125 = \gamma_v$$	(eq. G-1)

In a complex system such as a flight simulator, the system architect must be aware of the gamma at every stage of the system, starting from the source of the imagery (e.g. camera or satellite) right through to the simulator's display device. His objective is to ensure that product of all gammas match the viewing gamma of the simulator.

Given the above assumptions, and our objective of ensuring that the product of all gammas in the viewing chain equals the viewing gamma, the modeler will end up (subjectively) adjusting images to an equivalent file gamma of 1.25.

The bottom portion of the illustration show the path taken by the CDB imagery as it is ingested first by the real-time publisher, then by the IG, the IG color look-up tables and finally through to the visual display system. In this example, we will assume the following:

1. The imagery file in the CDB is unmodified (i.e. those produced by the Adobe Photoshop at the DBGF). Note that as a result of viewing gamma of γ_v = 1.25, the file gamma ended up at γ_f = 1.25 at the DBGF. As a result, the CDB also has a file gamma of γ_f = 1.25

2. The IG performs all of its internal operations in a linear color space (i.e. the IG_gamma is γ_{IG} = 1.00)
3. The simulator visual system produces an average scene brightness of approximately 6 ft-lamberts: under these viewing conditions, the viewing gamma is γ_{v} = 1.125.
4. The measured gamma of the visual display system is γ_{crt} = 2.025
5. The content of the IG's color look-up tables is adjusted to compensate for the gamma of the visual display system, i.e. it is loaded with γ_{lut} = (1/2.025)

Given the above assumptions, and our objective of ensuring that the product of all gammas in the chain equals the viewing gamma of 1.125, the required visual run-time publisher gamma must account for the difference in viewing gamma at the DBGF and at the simulator. As a result, the publisher gamma must be (1.125/1.25).

6.6.2. Harmonization of Gamma at DBGF with Gamma of Simulator Visual System

Both the modelers and the visual system architects should be keenly aware of the handling of gamma at the Data Store Generation Facility and at the simulator. Figure G-1: Typical Handling of Gamma at DBGF and Simulator, illustrates the typical handling of gamma in both of these cases.

The top portion of the illustration shows the path taken by source data as a modeler is viewing it at this workstation via the application software. In this example, we will assume the following:

1. The DBGF imagery application is Adobe Photoshop. The default color space profile used by Adobe Photoshop (i.e. the *.icm file) is the sRGB Color Space Profile which is defined by the sRGB standard to be a gamma of 2.2, therefore the Photoshop uses a γ_{lut} = (1/2.2)
2. The DBGF workstation is running Windows (therefore the O/S does not gammatize the imagery before sending it to the display, γ_{lut} = 1.25)
3. The measure gamma of the DBGF workstation monitor is γ_{crt} = 2.2
4. The DBGF workstation is located in a dimly lit room, so the viewing gamma is in effect γ_{v} = 1.25

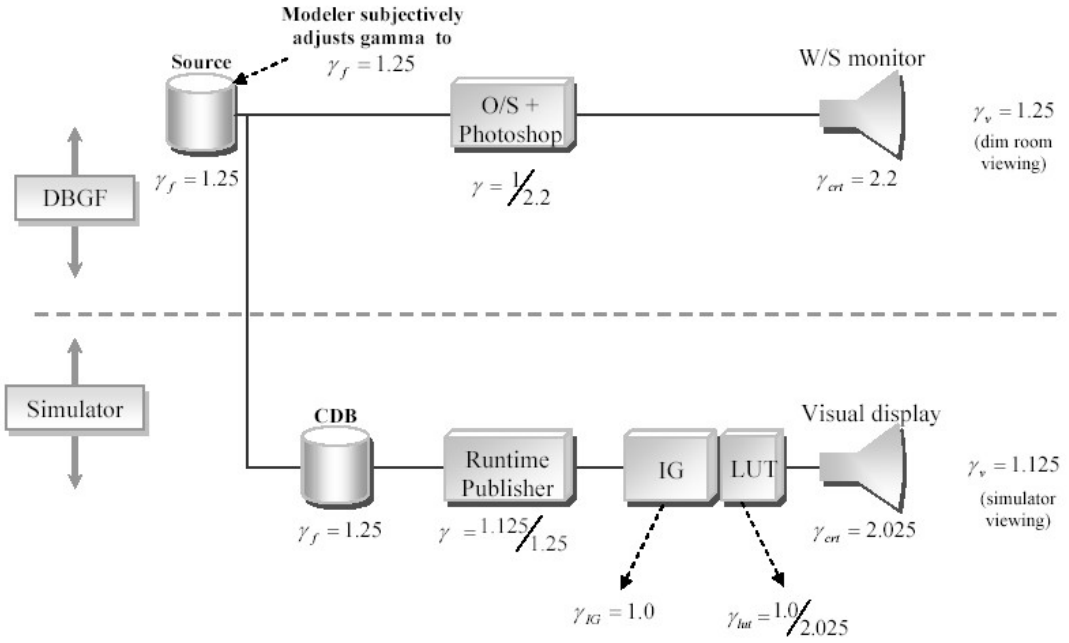


Figure G-1: Typical Handling of Gamma at DBGF and Simulator

6.7. Handling of Color

The default CDB standard color space follows the same convention as the Windows sRGB Color Space Profile. *sRGB* is the default color space in Windows, based on the IEC 61966-2-1 Standard. A *sRGB* compliant device does not have to provide a profile or other support for color management to work well.

Nonetheless, whether calibrated or not to the IEC Standard, all variants of RGB are typically close enough that undemanding viewers can get by with simply displaying the data without color correction. By storing calibrated RGB, the CDB standard retains compatibility with existing database tools and software programs that expect RGB data, yet provides enough information for conversion to XYZ in applications that need precise colors. Thus, the CDB standard gets the best of both worlds.

Full compliance to the CDB standard requires adherence to the color space described in this section. However, in virtually all cases, direct use of un-calibrated RGB is sufficient. The builders of Synthetic Environment Databases and the users of Visual Systems should be aware of these color space conventions; significant deviation from the underlying IEC assumptions may yield significant color differences.

The CDB standard encoded RGB color tri-stimulus values assume the following:

1. Display luminance level: 80 cd/m2
2. Display white point x = 0.3127, y = 0.3291 (D65)
3. Display model Offset (R, G and B): 0.055
4. Display Gun/Phosphor Gamma (R, G, and B): 2.2

Table G-1. CIE Chromaticity for CDB Reference Primaries & CIE Standard Illuminant

	Red	Green	Blue	D65 (white)
X	0.6400	0.3000	0.1500	0.3127
Y	0.3300	0.6000	0.0600	0.3291
Z	0.0300	0.1000	0.7900	0.3583

According to PNG (Portable Network Graphics) Specification Version 1.0, W3C Recommendation 01-October-1996 Appendix, Color Tutorial,

(http://www.w3.org/TR/PNG-GammaAppendix):

“The color of an object depends not only on the precise spectrum of light emitted or reflected from it, but also on the observer, their species, what else they can see at the same time, even what they have recently looked at. Furthermore, two very different spectra can produce exactly the same color sensation. Color is not an objective property of real-world objects; it is a subjective, biological sensation. However, by making some simplifying assumptions (such as: we are talking about *human* vision) it is possible to produce a mathematical model of color and thereby obtain good color accuracy.

6.7.1. Device-dependent Color

Display the same RGB data on three different monitors, side by side, and you will get a noticeably different color balance on each display. This is because each monitor emits a slightly different shade and intensity of red, green, and blue light. RGB is an example of a device-dependent color model; the color you get depends on the device. This also means that a particular color represented as say RGB 87, 146, 116 on one monitor might have to be specified as RGB 98, 123, 104 on another to produce the *same* color.

6.7.2. Device-independent color

A full physical description of a color would require specifying the exact spectral power distribution of the light source. Fortunately, the human eye and brain are not so sensitive as to require exact reproduction of a spectrum. Mathematical, device-independent color models exist that describe fairly well how a particular color will be seen by humans. The most important device-independent color model, to which all others can be related, was developed by the International Commission on Illumination in 1931 (CIE-1931, in French) and is called “CIE XYZ” or simply “XYZ.”

In XYZ, X is the sum of a weighted power distribution over the whole visible spectrum. So are Y and Z, each with different weights. Thus any arbitrary spectral power distribution is condensed down to just three floating-point numbers. The weights were derived from color matching experiments done on human subjects in the 1920s. CIE XYZ has been an International Standard since 1931, and it has a number of useful properties:

1. Two colors with the same XYZ values will look the same to humans
2. Two colors with different XYZ values will not look the same
3. The Y value represents all the brightness information (luminance)
4. The XYZ color of any object can be objectively measured

Color models based on XYZ have been used for many years by people who need accurate control of color i.e., lighting engineers for film and TV, paint and dyestuffs manufacturers, and so on. They are thus proven in industrial use. Accurate, device-independent color started to spread from high-end, specialized areas into the mainstream during the late 1980s and early 1990s, and CDB takes notice of that trend.

6.7.3. Calibrated, Device-Dependent Color

Traditionally, image file formats have used uncalibrated, device-dependent color. If the precise details of the original display device are known, it becomes possible to convert the device-dependent colors of a particular image to device-independent ones. Making simplifying assumptions, such as working with CRTs (which are much easier than printers), all we need to know are the XYZ values of each primary color and the CRT exponent.

So why does not the CDB standard store images in XYZ instead of RGB? Well, two reasons. First, storing images in XYZ would require more bits of precision, which would make the files bigger. Second, all programs would have to convert the image data before viewing it. But more importantly, whether calibrated or not, all variants of RGB are close enough that undemanding viewers can get by with simply displaying the data without color correction. By storing calibrated RGB, the CDB standard retains compatibility with existing database tools and software programs that expect RGB data, yet provides enough information for conversion to XYZ in applications that need precise colors. Thus, we get the best of both worlds.

6.8. What are chromaticity and luminance?

Chromaticity is an objective measurement of the color of an object, leaving aside the brightness information. Chromaticity uses two parameters x and y, which are readily calculated from XYZ:

$x = X/(X + Y + Z)$ $y = Y/(X + Y + Z)$	(eq. G-3)

XYZ colors having the same chromaticity values will appear to have the same hue but can vary in absolute brightness. Notice that x,y are dimensionless ratios, so they have the same values no matter what units we've used for X,Y,Z.

The Y value of an XYZ color is directly proportional to its absolute brightness and is called the luminance of the color. We can describe a color either by XYZ coordinates or by chromaticity x,y plus luminance Y. The XYZ form has the advantage that it is linearly related to RGB intensities.

6.9. How are computer monitor colors described?

The "white point" of a display device is the chromaticity x,y of the monitor's nominal white, that is, the color produced when R = G = B = maximum.

It's customary to specify CRT monitor colors by giving the chromaticities of the individual

phosphors R, G, and B, plus the white point. The white point allows one to infer the relative brightness of the three phosphors, which isn't determined by their chromaticities alone.

NOTE: The absolute brightness of the monitor is not specified. For computer graphics work, we generally don't care very much about absolute brightness levels. Instead of dealing with absolute XYZ values (in which X,Y,Z are expressed in physical units of radiated power, such as candelas per square meter), it is convenient to work in "relative XYZ" units, where the monitor's nominal white is taken to have a luminance (Y) of 1.0. Given this assumption, it's simple to compute XYZ coordinates for the monitor's white, red, green, and blue from their chromaticity values.

6.10. How do I convert from source_RGB to XYZ

Make a few simplifying assumptions first, like the monitor really is jet black with no input and the guns don't interfere with one another. Then, given that you know the

CIE XYZ values for each of red, green, and blue for a particular monitor, you put them into a matrix M:

$$M = \begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix} \quad \text{(eq. G-4)}$$

RGB intensity samples normalized to the range zero to one can be converted to XYZ by matrix multiplication.

NOTE	If you the RGB samples are gamma-encoded, the gamma encoding must be un-done.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = M \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad \text{(eq. G-5)}$$

In other words, X = Xr*R Xg*G Xb*B, and similarly for Y and Z. You can go the other way too:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = M^{-1} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 3.2410 & -1.5374 & -0.4986 \\ -0.9692 & 1.8760 & 0.0416 \\ 0.0556 & -0.2040 & 1.0570 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \text{(eq. G-6)}$$

Where M^{1}= The inverse of the matrix *M* used to go from XYZ-1931 color space to the CDB specification RGB color space.

In the RGB encoding process, negative sRGB tri-stimulus values, and sRGB tri-stimulus values greater than 1,00 are not retained. When encoding software cannot support this extended range, the luminance dynamic range and color gamut of RGB is limited to the tri-stimulus values between 0,0 and 1,0 by simple clipping.

According to PNG (Portable Network Graphics) Specification Version 1.0, W3C Recommendation 01-October-1996 Appendix, Color Tutorial,

(http://www.w3.org/TR/PNG-GammaAppendix):

“The gamut of a device is the subset of visible colors that the device can display. (Note that this has nothing to do with gamma.) The gamut of an RGB device can be visualized as a polyhedron in XYZ space; the vertices correspond to the device’s black, blue, red, green, magenta, cyan, yellow, and white.

Different devices have different gamut (e.g. database generation workstation, simulator display systems). In other words one device may be able to display certain colors (usually highly saturated ones) that another device cannot. The gamut of a particular RGB device can be determined from its R, G, and B chromaticities and white point.

Converting image data from one device to another generally results in gamut mismatches colors that cannot be represented exactly on the destination device. The process of making the colors fit, which can range from a simple clip to elaborate nonlinear scaling transformations, is termed gamut mapping. The aim is to produce a reasonable visual representation of the original image.

Chapter 7. ShapeFile dBASE III guidance

Was B.1.3 Annex B Volume 2 of the OGC CDB Best Practice. Now in Volume 4: OGC CDB Rules for Encoding CDB Vector Data using Shapefiles (Best Practice).

Chapter 8. TIFF Implementation Guidance

Formerly Annex B in the OGC CDB Best Practices.

The conventions define how TIFF files are interpreted by a CDB-compliant TIFF reader; the stated conventions supersede or replace related aspects of this annotated standard. Unless stated otherwise, CDB-compliant TIFF readers shall/will ignore any data that fails to conform to the stated conventions. Please refer to TIFF 6.0 (1992).

Section 2 TIFF Structure	CDB-compliant TIFF readers do not consider TIFF image and DEM data in big-endian byte order.
Multiple Images per TIFF File (16)	*TIFF/CDB Readers:* The CDB Standard does not take advantage of multiple images per TIFF file.
Section 3: Bi-level images (17)	CDB-compliant TIFF readers do not consider bi-level image data.
Color (17)	CDB-compliant TIFF readers do not consider WhiteIsZero image data
Compression (17)	CDB-compliant TIFF readers do not consider compressed TIFF image.
Rows and Columns (18)	The CDB standard recommends that the product of ImageLength and ImageWidth be less than 4,194,304 (2K x 2K).
	CDB-compliant TIFF readers require this field to be a power of 2. ImageLength need not be equal to ImageWidth
	CDB-compliant TIFF readers require this field to be a power of 2. ImageWidth need not be equal to ImageLength
ResolutionUnit (18)	CDB-compliant TIFF readers do not consider this TIFF tag
XResolution (19)	CDB-compliant TIFF readers do not consider this TIFF tag.
YResolution (19)	CDB-compliant TIFF readers do not consider this TIFF tag.
Bi-level Compression (22)	CDB-compliant TIFF readers do not consider compressed TIFF image.
Palette Color Images (23)	CDB-compliant TIFF readers do not consider palette-color images, i.e. PhotometricInterpretation = 3 (Palette Color)
ColorMap (23)	CDB-compliant TIFF readers do not consider palette-color images; as a result, this tag is ignored.

General Requirements (26)	CDB-compliant TIFF readers consider only type 'II', (little-endian) byte ordered data.
	The CDB standard does not take advantage of multiple images per file.
Notes on Required Fields (28)	CDB-compliant TIFF readers require ImageWidth and ImageLength fields to be a power of 2. ImageWidth need not be the same as ImageLength. CDB-compliant TIFF readers do not consider data that does not conform to this requirement.
	CDB-compliant TIFF readers do not consider the XResolution and YResolution TIFF tags.
	CDB-compliant TIFF readers do not consider the ResolutionUnit TIFF tag.
Artist Tag (29)	CDB-compliant TIFF readers do not consider the Artist TIFF tag.
CellLength (29)	CDB-compliant TIFF readers do not consider the CellLength TIFF tag. The length of the dithering or halftoning matrix used to create a dithered or halftoned bilevel file. Tag = 265 (109.H) Type = SHORT N = 1 This field should only be present if Threshholding = 2 No default. See also Threshholding.
CellWidth (29)	CDB-compliant TIFF readers do not consider the CellWidth TIFF tag
ColorMap (29)	CDB-compliant TIFF readers do not consider the ColorMap TIFF tag
Compression (30)	CDB-compliant TIFF readers do not consider compressed TIFF image
Copyright Tag (31	CDB-compliant TIFF readers do not consider the Copyright TIFF tag
DateTime Tag (31)	CDB-compliant TIFF readers do not consider the DateTime TIFF tag
Unassociated alpha data (32)	CDB-compliant TIFF readers do not consider unassociated alpha data.
FreeByteCounts Tag (33)	CDB-compliant TIFF readers do not consider the FreeByteCounts TIFF tag
FreeOffsetsTag (33)	CDB-compliant TIFF readers do not consider the FreeOffsets TIFF tag.
GreyResponseCurve Tag (33)	The CDB standard assumes that the gray response curve of TIFF image files to be linear. As a result, CDB-compliant TIFF readers do not consider the GrayResponseCurve Tag data.

GreyResponseUnit (33)	The CDB standard assumes that the gray response curve of TIFF image files to be linear. As a result, CDB-compliant TIFF readers do not consider the GrayResponseUnit Tag data.
HostComputer Tag (34)	CDB-compliant TIFF readers do not consider the HostComputer TIFF tag
Image Description Tag (34)	CDB-compliant TIFF readers do not consider the HostComputer TIFF tag
ImageLength (34)	CDB-compliant TIFF readers require ImageWidth and ImageLength fields to be a power of 2. ImageLength need not be the same as ImageWidth. CDB-compliant TIFF readers do not consider image that does not conform to this requirement.
Make Tag (35)	CDB-compliant Tiff readers do not consider the Make TIFF tag.
MaxSampleValue Tag (35)	The CDB standard establishes that the MaxSampleValue to be always equal to the maximum value that can be represented by the number format representation. As a result, CDB-compliant TIFF readers do not consider the MaxSample TIFF tag
MinSampleValue (35)	The CDB standard establishes that the MinSampleValue to be always equal to 0 for image data and to minimum value that can be represented by the number format representation. As a result, CDB-compliant TIFF readers do not consider the MinSample TIFF tag.
Model (35)	CDB-compliant TIFF readers do not consider the Model TIFF tag.
NewSubfileType (36)	The CDB standard assumes that the data is full or reduced resolution only. As a result, CDB-compliant TIFF readers only consider images and DEMs data whose PhotometricInterpretation Tag value is 1 or 2.
Orientation (36)	The CDB standard assumes that the data is organized such that the 0th row represents the visual top of the grid data (or image), and the 0th column represents the visual left-hand side. As a result, CDB-compliant TIFF readers do not consider image and DEM data whose Orientation Tag value is not 1
Photometric Interpretation (37)	CDB-compliant TIFF readers only consider images whose PhotometricInterpretation Tag value is 1 or 2.
	CDB-compliant TIFF readers do not consider WhiteIsZero images

	CDB-compliant TIFF readers do not consider palette-color images, i.e. PhotometricInterpretation = 3 (Palette Color).
	CDB-compliant simulator TIFF readers do not consider transparency mask imagery data, i.e. PhotometricInterpretation = 4 (Transparency Mask).
ResolutionUnit Tag (38)	The CDB standard establishes a series of conventions that govern the resolution of TIFF files. As a result, CDB-compliant TIFF readers do not consider the ResolutionUnit TIFF tag.
Software Tag (39)	CDB-compliant TIFF readers do not consider the Software TIFF tag.
SubFileType Tag (41)	CDB-compliant TIFF readers do not consider image whose Subfile type = 3.
Thresholding Tag (41)	CDB-compliant TIFF readers do not consider image data whose Thresholding TIFF tag is not equal 1
XResolution Tag (41)	The CDB standard establishes a series of conventions that govern the resolution of TIFF files. As a result, CDB-compliant TIFF readers do not consider this TIFF tag
YResolution Tag (41)	The CDB standard establishes a series of conventions that govern the resolution of TIFF files. As a result, CDB-compliant TIFF readers do not consider this TIFF tag.
Section 9: Packbits (42)	CDB-compliant TIFF readers do not consider PackBits compressed TIFF data. As a result, section 9 is not applicable to CDB-compliant TIFF readers.
Section 10: Huffman Compression (43)	CDB-compliant TIFF readers do not consider Modified Huffman compressed TIFF data. As a result, section 10 is not applicable to CDB-compliant TIFF readers.
Section 11 CCIT Bilvel Encodings (49)	CDB-compliant TIFF readers do not consider CCITT Bi-level encoded TIFF data. As a result, section 11 is not applicable to CDB-compliant TIFF readers.
Section 12: Document Storage and Retrieval (55)	CDB-compliant TIFF readers do not consider all TIFF tags related to document storage and retrieval. As a result, section 12 is not applicable to CDB-compliant TIFF readers.
Section 14: Differencing Predictor (64)	CDB-compliant Tiff readers do not consider Differencing Predictor compressed TIFF data. As a result, section 14 is not applicable to CDB-compliant TIFF readers.

Section 16: CMYK Images (69)	CDB-compliant TIFF readers do not consider CMYK encoded color image TIFF image data. As a result, section 16 is not applicable to CDB-compliant TIFF readers.
Section 17: Halftone hints (72)	CDB-compliant TIFF readers do not consider any of the TIFF tags related to halftone hints. As a result, section 17 is not applicable to CDB-compliant TIFF readers.
Unassociated Alpha and Transparency masks (78)	CDB-compliant TIFF readers do not consider unassociated alpha image data.
Section 19: Sample Data format (80)	The CDB standard establishes the conventions that govern the SampleFormat of TIFF image and DEM data. As a result, CDB-compliant TIFF readers do not consider image and DEM data when the value of the SampleFormat tag does not conform to CDB conventions. While the above-mentioned sample data formats are possible, CDB clients expect image and DEM data to be in the format as specified in the CDB conventions and constraints.
SMinSampleValue Tag (80)	CDB-compliant TIFF readers do not consider the SMinSampleValue TIFF tag.
SMazSampleValue (81)	CDB-compliant TIFF readers do not consider the SMaxSampleValue TIFF tag.
Section 20: RGB Image Colorimetry (82)	CDB-compliant TIFF readers do not consider any of the TIFF tags describes in this section.
Section 21 YC-C Images (89)	CDB-compliant TIFF readers do not consider YC $_{b}C_{r}$ color encoded TIFF image data. As a result, section 21 is not applicable to CDB-compliant TIFF readers.
Section 22: JPEG Compression (95)	CDB-compliant TIFF readers do not consider JPEG color encoded TIFF image data. As a result, section 22 is not applicable to CDB-compliant TIFF readers.
Section 23: CIE Lab Images (110)	CDB-compliant TIFF readers do not consider CIE L*a*b* color encoded TIFF image data. As a result, section 23 is not applicable to CDB-compliant TIFF readers.

Annex A: Revision History

Date	Release	Editor	Primary clauses modified	Description
2016-04-04	1.0	C. Reed		OAB review version
2016-06-25	R2	C. Reed	Various	Prepare for adoption vote. Incorporate changes based on public comment period.
2016-10-05	R3	C. Reed	Various	Preparation for publication
2016-11-22	1.0	C. Reed		Ready for publication
2017-12-28	1.1	C. Reed	Various	Minor edits for version 1.1.
2019-12-16	1.2	C. Reed	Various	Changes for version 1.2

Annex B: Bibliography