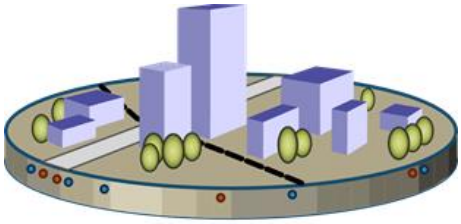
Title: 3D Cities Information Model Overview
Version: 1.3 Beta
Date: March 28, 2014

Introduction

This document explains the purpose and meaning of the Feature Classes, Object tables, Domains and Relationships that make up the 3D Cities Information Model (3DCIM).

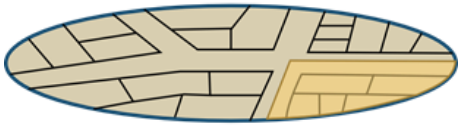
A 3D City is a collection of features, networks and surfaces, but the data structure can be simplified by grouping them into some basic themes: the **Built Environment**, the **Legal Environment**, and the **Natural Environment**. Each of these themes shares some common attributes and traits, which are described below.

The Built Environment



The Built Environment is comprised of features and networks that are created or actively managed by humans. These features include: structures (buildings, bridges, tunnels), utility networks, multimodal transportation networks (interior & exterior), installations (e.g., street furniture, sensors), and street trees.

The Legal Environment



Features in the Legal Environment define land use plans and regulations and property ownership boundaries. These include Land Use Zones, which can have a nested structure (zones that are within and override the regulations of larger zones), and may have both 2D and 3D dimensional attributes, like maximum buildable heights. These regulations are typically stored as tables and may also apply to Parcel (ownership) boundaries.

The Natural Environment

The Natural Environment is comprised of all naturally occurring features on, above, or below the earth's surface. This can include the land cover (wilderness areas, biomes, water bodies), but also surface/subsurface geologic structure and above-earth atmosphere, climate, and weather.

Design Principles

The 3DCIM is designed to be compact and simple in its structure, making the core of the model easy to understand and to populate with data. The following principles are applied in in its development.

Simplicity

The 3DCIM should be compact and simple. The 3DCIM is not to replace existing models such as the Local Government Model[1] or the BISDM[2]. It should reflect the core data sets that cities throughout the world have available and represent the structures that are essential for high-priority use cases specific to 3D City applications. This is also important since

[1] Please refer to http://www.arcgis.com/home/item.html?id=ae175b36c4154dda987127dff879350d

[2] Please refer to http://support.esri.com/en/downloads/datamodel/detail/44

data models tend to grow substantially over time, and while it is easy to add something, it can be next to impossible to remove something.

Compatibility

The 3DCIM has to be compatible to major standards that are used by urban communities, specifically CityGML 1.0, the Esri Local Government Data Model, INSPIRE and FGDC-STD-003. Compatibility means that those aspects that are shared between the standard and the 3DCIM have to be modeled as free of mismatches as possible, and that the 3DCIM contains enough information so that valid structures comparable to those imported can also be exported again (roundtrip interoperability).

Extensibility and Internationalization

The 3DCIM is a part of a solution template. As with other templates, this template represents a collection of best practices and provides a harmonized data model for core aspects of a 3D city. We fully expect users to extend and to localize the existing model, and should make this an easy process. Furthermore, we expect the 3DCIM to evolve substantially over the next few years, so we should make our own work easier as well. Thus, the model makes use of related object or feature classes at each point where extensibility is likely to be required.

Expressivity

The 3DCIM needs to be expressive, i.e. its constructs need to be specific enough to be of high direct utility in the applications. This principle is often in conflict to the extensibility and simplicity principles, so a weighting has to be made in each case.

Model Content

This chapter describes the conceptual schema, organized by themes.

General Attributes

All feature classes have the following, generic attributes:

- `beginLifespan`: The point in time at which this feature starts to exist.
- `endLifespan`: The point in time at which this feature stops existing.
- `attribution`: Information on where this specific feature data came from.
- `name`: A natural name of the feature.
- `description`: A description of the feature.
- `status`: The status of the object (planned, underConstruction, built, underRemoval…)
- `[classname]FID`: A resolvable feature ID.

Feature IDs (FIDs) are unique across a 3DCIM database and use the following pattern:

```
FeatureClassName/[type/]LocalID[/Version]
Building/Residential/1245/4
```

The reason for this construct is that it enables flexible linking both in the database and in applications, such as a HTML5/JavaScript web app. As an example, a `Regulation` object can be related to a `ZoningDistrict` or to a `Parcel`. FIDs using this format can be easily used together with a namespace URL to provide feature identifiers that are globally resolvable via the network, enabling the creation of 3D mashups.

Furthermore, almost all feature classes and some of the object classes have two fields that follow the `type`/`subtype` pattern. The `type` is usually bound to a code value domain. This reliance on a known set of values can help greatly in defining rules and procedural representations. The `subtype` field is usually a simple string field, which may be set to any value, allowing for simple extensibility.

AnnotatedSpace, Association, AttributeContainer

The 3DCIM has several generic classes that allow its adoption for many applications without having to modify the model too much. The first is the `AnnotatedSpace`. An `AnnotatedSpace` is, compared to the standard Annotation class, not a label with placement and styling information. Instead, it serves as a container for adding information to any location or area within the 3d city model - on a terrain or building surface or even on a certain volume. Such an annotation can be used to store comments, discussions, notes, processing results, links to additional resources and other things. It can be seen as a Point of Interest object that can be used for many different applications.

Figure 1: An example spatial annotation tool used to highlight features on a wall

Optionally, an `AnnotatedSpace` can also be associated with any other feature in the city, such as a `Building` or a `TransportNetworkElement`.

The second generic class makes the model extensible and keeps its complexity low. The `CityFabricRelation` is a generic, non-enforced mechanism of storing associations between Features in a 3D cities model. Via this mechanism, dynamic associations such as aggregations or *partOf* relations, multiple representations, or spatial relations can be maintained. This mechanism is used in addition to the geodatabase's capabilities where loosely coupled associations are required. A disadvantage of using them is that the DBMS cannot guarantee referential integrity and that for some query types, multiple steps have to be made.

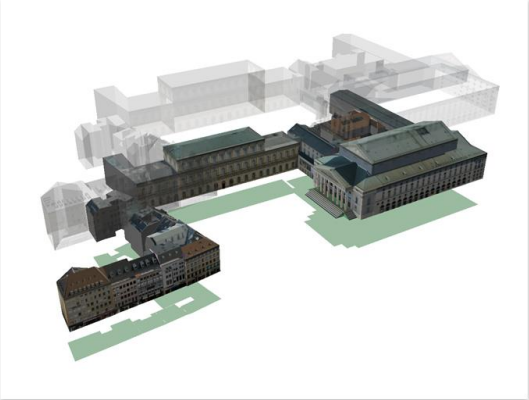
The third generic class is called `AttributeContainer` and is used to transport original attributive data into the 3DCIM. This is needed when editing features such as buildings and having to maintain its properties. An example for this is importing a CityGML model into the 3DCIM, editing it in CityEngine, and then exporting it again to CityGML, complete with semantics and attached attributes.

Built Environment

The Built Environment of a city comprises all physical structures and logical networks that are man-made. Features in the built environment are generally time-enabled.

Buildings

Buildings are the central features of most urban environments. In the scope of the 3DCIM, a `Building` feature has a set of attributes that describe its outer form and a footprint polygon, such as total height, eave height, roof type, number of levels above and below ground and for recording information on its type and usage. Its 3D representation[3] is stored in the `BuildingShell` and `BuildingShellPart` feature classes. Each `Building` may have one or more `BuildingShells`, typically from different sources such as Pictometry data, hand-modeled or procedurally created. `BuildingShells` are watertight and complete representations of the entire building. For some applications, such a model is not ideally suited, so the 3DCIM also allows the usage of `BuildingShellParts`. These can be used in addition or instead of a `BuildingShell` to separately store walls, roofs or other significant parts of a building's outer shell. To denote the role of a `BuildingShellPart`, a type attribute is used that can be set to `complete`, `wall`, `roof` or `ground`. Both shells and shell parts have attributes denoting the generator that created them and a flag attribute called `protectionLevel`, which can be used to keep representations from being overwritten e.g. by procedural representations.



A `Building` has several connected features, which represent entrances using Point features of the type `BuildingEntrancePoint` and the ridgelines using a Polyline feature called `BuildingRidgeLine`.

Interiors

The interior spaces of a city are at least as important to the residents as the exterior spaces. The interior model of the 3DCIM focuses on spaces in buildings, their functions and the transport networks that connect those spaces. It is closely related to the interiors model of the Local Government Information Model.

The core feature class of the interior model is called a `BuildingInteriorSpace`. A `BuildingInteriorSpace` can be a room or a part of a room, or a duct/shaft passing multiple levels in a building. It can also be a volume that includes more than one room, and is also not bound to a single floor. The subtype field of a space feature can be freely defined, but should make use of established classification systems such as the Open Standards Consortium for Real Estate (OSCRE).

A `BuildingFloor` is an organizational structure within a `Building`. A single `BuildingFloor` Feature may have multiple parts, such as different, connected or unconnected parts at the same building level. A `BuildingFloor` does not represent a detailed floor plan – instead, structures on a floor are modeled using `BuildingInteriorStructures`. These represent walls and other physical structures such as doors in the building and are always linked to a `Floor`. `BuildingInteriorStructures` have attributive information attached to them, such as their height and the material they are made of.

[3] Corresponding to LOD1 to LOD3 in CityGML

InteriorInstallations include a wide range of devices such as HVAC units, copiers or shelves, tables and much more. The concrete usage of features of this class depends on the application. Usually, InteriorInstallations are linked to a BuildingInteriorSpace.

Finally, the Transport Network relates closely to the Interior Model and has associations to BuildingInteriorSpaces. There are several specific types for interior transport network elements, such as Corridor, Elevator, Ramp or Stairs.

Installations

Installations refer to different types of objects in public space, campuses or inside buildings. This group includes features such as street furniture, sensors and signs. All types are Point features and have rotational information in a field.

StreetFurniture is a collective term for objects and pieces of equipment installed on streets and roads for various purposes. It includes benches, traffic barriers, bollards, post boxes, phone boxes, streetlamps, bus stops, tram stops, fountains, public sculptures, information signs and waste receptacles. Traffic lights and traffic signs fall into a separate feature class called Sign and are linked to at least one TransportNetworkElement. For the type attribute, valid values are defined in the StreetFurnitureType domain from the Local Government Information Model.

A Sensor can represent a wide range of different sensors, from meteorological stations to video surveillance cameras or motion detectors. A sensor feature has a sensor type that indicates a general classification of the phenomenon it observes, a feed format (e.g. as a MIME type) and a feed type, which indicates periodicity and other properties of the stream or feed that this sensor produces. The sensor object furthermore contains information as to the location from which its data can be obtained.

Finally, a Sensor has a SensorCoverage, which is a Polygon feature class indicating the area which the sensor produces readings for.

Transport Networks

Different types of Transport Networks, such as roads, railways and footpaths, become more and more interleaved. Especially in cities, reacting to a dynamic environment enables traffic participants to choose their mode of transport, going from one to another as it fits current needs and as it can be accommodated by the urban environment. Indoor and outdoor networks are explicitly linked, and height is maintained for all elements of the network.

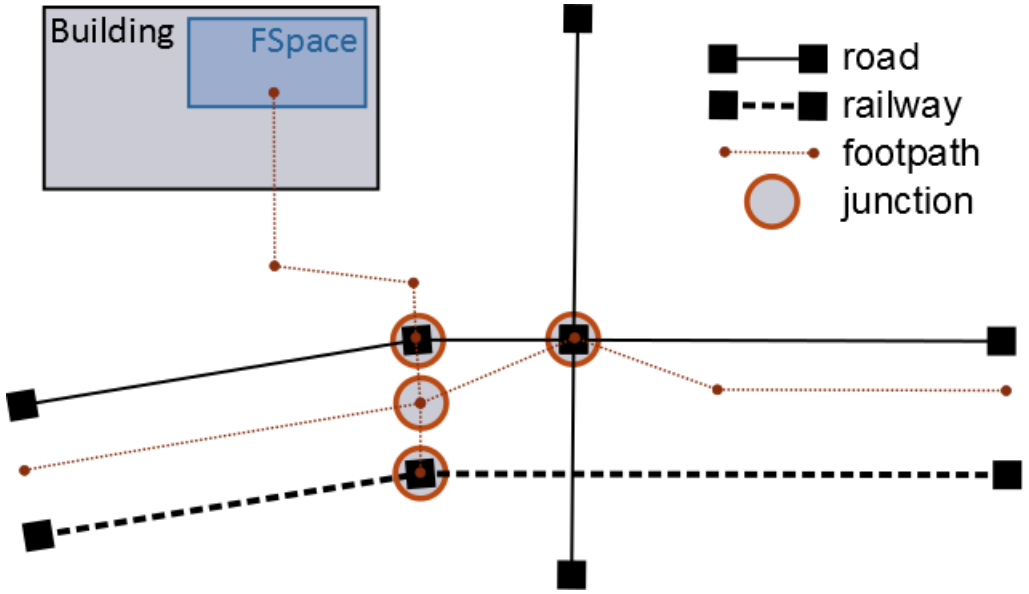
The Transport Network within the 3DCIM is modeled as a Network Dataset[4] containing edges, junctions and optionally, turns. TransportNetworkElement features represent the edges of the network. Each TransportNetworkElement represents a segment of a single type, such as a road, railway, ferry line, bicycle lane, sidewalk or indoor corridor. Just a single feature class is used for all types of transport, reducing the number of feature classes and again emphasizing the connected nature of transport networks in a city. It is recommended to model individual lanes only if they are physically separated,

[4] As recommend for transport use cases in “Understanding Geometric Networks” by Erik Hoel, Craig Gillgrass, 2011.

e.g. with a barrier or a grass strip. This enables an improved generic generation process where users can make use of a road profile editor easily to control the generation of procedural representations of a part of the transport network.

For these `TransportNetworkElements`, standard fields include allowed travel direction (forward, backward, both), a name, the hierarchy level used for solving, the semantic classification and preference information for various modes of transport. The latter are managed in an object class named `TransportElementPreference` and relate a transport mode to a segment type and a preference code such as *Prohibit*, *Avoid: High/Med/Low* and *Prefer:High/Med/Low*.

`TransportNetworkNodes` are the features that represent junctions in the network. At such a junction, an agent travelling the network can change the mode of transport if two heterogeneous `TransportNetworkElements` connect. No fixed assumptions are made whether a given mode can directly connect to another (e.g. car to train) – this should be decided during network data import by the users. A `TransportNetworkNode` has several attributes that describe its form in a more precise manner, such as a type (e.g. roundabout, crossroads, and clover-leaf) and a freely definable subtype.

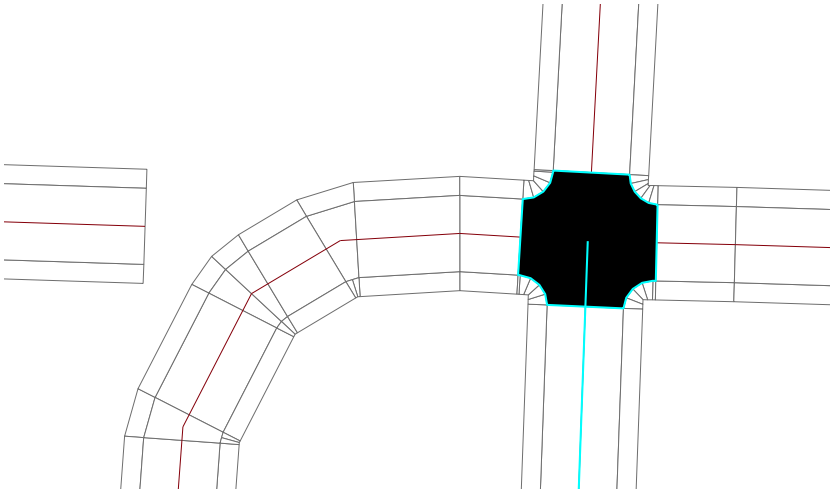


An example multimodal Transport Network

In the example network above, a person can travel by car on the road, walk into the building to do something in the `BuildingInteriorSpace` (such as buying a train ticket), and walk out again to the station to take the train towards the final destination. For sidewalks, cycle lanes and roads, connecting junctions should be placed wherever there is a cyclist or pedestrian crossing of any type.

All `TransportNetworkElements` may have one or multiple `NetworkElementRepresentation` objects. Such representation objects are modeled in a similar fashion as `BuildingShells`, but with a notable difference: A single `NetworkElementRepresentation` can either be generated from a single `TransportNetworkElements`, or from multiple ones e.g. on a node. The Figure below shows two examples for this:

1. The selected crossing's shape is dependent on the four incoming road segments. It represents a part of each of these segments and is thus a shared representation.
2. On the left side, there are nodes in the road which lead to additional shapes being generated for purposes of improving the geometric quality of the representations. Note also that this depends on the angle between two road segments.



CityEngine Generated shapes form a street centerline network

Bridges and Tunnels are also part of the `TransportElementRepresentation` feature class. Bridges are often characteristic structures within a city, and more and more cities have large numbers of underground structures such as road or railway Tunnels. Within the 3DCIM, both mostly have a representational role, as the 3DCIM cannot replace the detailed CAD/BIM data models that are in use for maintenance and other tasks around such structures. A Tunnel usually needs holes in the terrain. It still has to be clarified how these holes can be created and persisted.

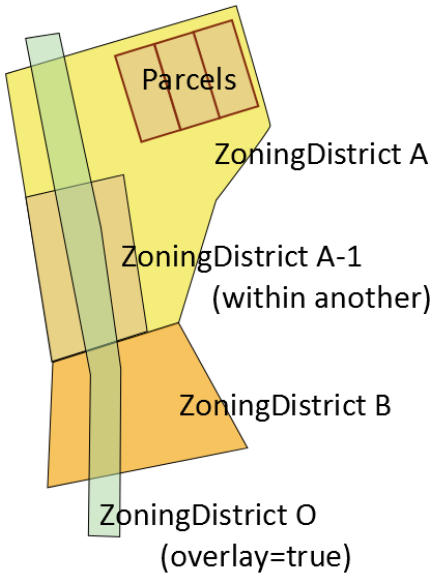
Trees and other individual vegetation objects

Trees and other significant, individual plants play an important role in the city and influence its look heavily. There are several typical processes in cities where trees play a role. However, at the same time, more often than not the actual data stored about a given tree is a very limited set. The model for trees thus focuses on a few core attributes such as type (genus) and subtype (such as a family or species), height, age and crown diameter.

It is recommended though that small plants such as flowers should not be handled individually, but should rather be generated on the base of the `LandCover` information.

Legal Environment

The legal environment represents ownership of urban spaces – not just parcels on the surface - and provides a detailed model for the maintenance of land use and zoning constraints.



Zoning

`ZoningDistricts` describe areas in different types of plans related to defining the current or future land-use and provide a full coverage of the area of interest. Plans that can be mapped to `ZoningDistricts` include development plans, land-use plans, comprehensive plans, regional plans or landscape plans, and can also be used to represent the land-use inventory. `ZoningDistricts` can be stacked in two ways:

1. If a `ZoningDistrict` is within another, any constraints defined in it override those defined in the outer `ZoningDistrict` or are added to the effective constraints.

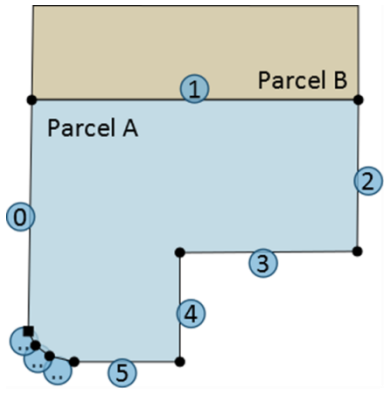
2. If a ZoningDistrict overlaps another ZoningDistrict that has the overlay attribute set to true, the constraints of the overlay ZoningDistrict are added to the list of valid constraints as alternatives. Overlay zones are usually used to allow additional land-uses, such as transport corridors or to designate multiple-use areas, or to indicate easements (right-of-way, flooding).

ZoningDistricts have a density attribute that is a unit-less float value in the range of 0 to 1. This value can be used to indicate high-density, medium-density or low-density zones. The zones also have optional fromZ and toZ attributes to indicate the vertical extent over the terrain for which the zone is valid.

The central aspect of a ZoningDistrict is the set of Regulations that are associated with it. Regulations have a constraintAspect tied to the ConstraintAspect domain (height, eaveHeight, siteCoverage, GFA, plotRatio, cubicIndex, levelsAboveGround/BelowGround, setbackFront/Side/Rear, skyviewAngle,…). They have a constraintOperator (min, max, avg, equal…) and a value that together define an exact limit for the given aspect. A Regulation can also have a borderCondition to indicate that this constraint is only valid if a parcel's border segment fulfills a specific topological condition, such as being adjacent to a street or to a parcel with a different land-use.

Parcels

Parcels represent ownership of land or of other spatial units and provide address and ownership information. They have a type attribute that indicates whether a Parcel feature shows land ownership, building ownership or unit ownership. Land ownership Parcels are the unit that zoning rules have to be applied to. Parcels can also be connected to Neighborhood objects, which describe the feature types that neighbor each segment of the parcel's border.



Attribute	Type
fromIndex	Integer
toIndex	Integer
parcelOID	Char(36)
connectedToType	Char(36)

Format for Attribute: FeatureClassName/Type (/SubType), e.g. TransportNetwork/Road/Local

A Parcel with Neighborhood information

Natural Environment

Land Cover

LandCover features are used to describe features of the natural landscape, like forests, waterbodies, wetlands and rock outcrops. They have a primaryLandCoverType and a secondaryLandCoverType attribute that both follow established main classifications for land cover (barren, cultivated, forest, shrub, standingWater, watercourse, wetland,…).

LandCover Features have a density attribute that indicates the portion of area taken up by the primaryLandCoverType. They share the subtype attribute with many other feature classes, which can be set freely to indicate specific types of cover, such as flowering plants. The final attribute, maxZExtent, provides the maximum height (positive value) or depth (negative value) of the feature.

A separate vegetation type is not available in the 3DCIM. Instead, if shrubs or tree groves are modeled, this should happen via an appropriate `LandCover` feature. Consider this example for the usage of primary and secondary land cover types:

```
primaryLandCoverType:    "forest",
secondaryLandCoverType:  "shrub",
density:                 0.6
maximumHeight:           35
```

This would indicate an area where 60% is covered with trees up to 35 (unit depends on vertical RS), while the remainder is covered with shrubs.

Basemap

The basemap is a group of coverage layers that represent the terrain and the texture of the terrain, usually using orthoimagery or a rasterized street map. It does not contain individual features, but rather provides a foundation for the other city features.

Terrain

Terrain can be represented in two potential ways as a basemap: As a raster layer or as a TIN layer. TINs have the advantage that distinct – and often surveyed – features such as breaklines of road dams are preserved well, while rasters can be managed very efficiently. However, depending on the raster's grid spacing, the raster sampling can lead to substantial omissions and errors. For the 3D City Model, the following data quality recommendations are made:

- *Raster*: Maximum horizontal spacing 10m, maximum vertical error at grid point 0.2m, maximum vertical error in between grid points 1.0 m.

Orthoimagery and Base Maps

For high-quality realistic visualization, orthoimagery draped onto the terrain makes a huge difference, especially if the quality of the orthoimagery is high.

An alternative to the use of high-resolution orthoimagery is to drape high-resolution rasterized base maps containing all elements of transport networks as well as basic land use/land cover information, but no text labels. If no orthoimagery or road map is available, alternatives include land cover maps, of hill shading or using color ramps to denote altitude differences.

Release Notes

The March 28, 2014 release is the first release of the 3D Cities Information Model for ArcGIS 10.2.

New Functionality

N/A

Resolved Problems

N/A

Known Issues

N/A