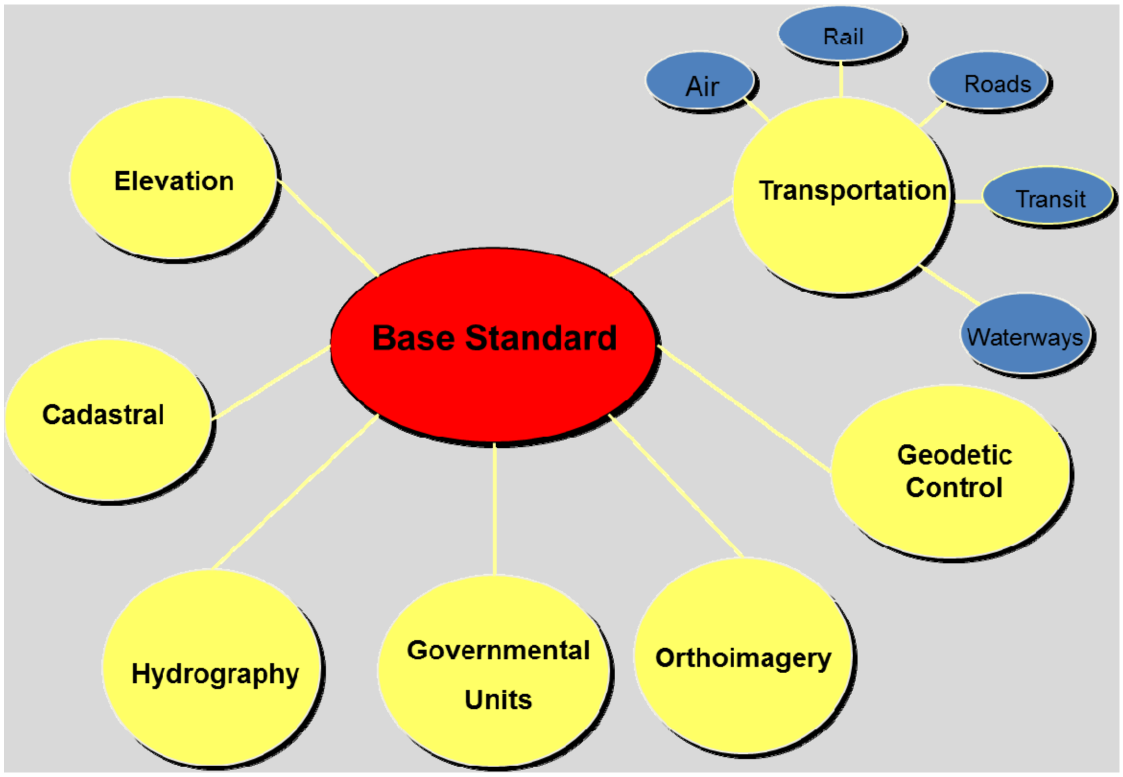
# Developing NSDI Framework Data: **ANSI Framework Data Content Standards** Guidance Document



**Version 1.0**

**December 2010**

# Developing NSDI Framework Data: ANSI Framework Data Content Standards Guidance Document

Version 1.0

December 2010

**Prepared for**
The Federal Geographic Data Committee
**By**
The Wyoming Geographic Information Science Center, University of Wyoming, Laramie, Wyoming USA

**Contributors**
Margo E. Berendsen, Wyoming Geographic Information Science Center
Jeffrey D. Hamerlinck, Wyoming Geographic Information Science Center
Shawn G. Lanning, Wyoming Geographic Information Science Center
With Assistance from
Sharon Shin, Federal Geographic Data Committee Secretariat Staff

The material presented herein resulted from research sponsored by the U.S. Federal Geographic Data Committee through Rocky Mountain Cooperative Ecosystem Study Unit Award #04121HS007 (1 September 2005 through 30 September 2010). Appreciation is extended to the following individuals for their insightful ideas, creative contributions and constructive criticism over the course of this project: Sharon Shin of the Federal Geographic Data Committee Secretariat Staff; Lynda Wayne, GeoMaxim; Phillip Dibner, OGC Interoperability Institute; and Steve Prager, University of Wyoming. Additional thanks goes to the volunteer reviewers of the draft version of this document and the associated Framework Standard training materials also developed under this scope of work.

**Note:** This guidance document is designed to be used in conjunction with the published, approved sections of the ANSI Framework Data Content Standards. Its primary purpose is to providing further clarification and explanations for understanding, implementing and teaching about the Framework Data Content Standards.

## Table of Contents

## List of Figures

## List of Tables

## Executive Summary

In a broad sense, *framework* can best be described as a basic conceptual structure used to solve complex issues. The Framework component of the U.S. National Spatial Data Infrastructure refers specifically to those geospatial data themes identified as the *foundation* upon which all other data layers are structured and integrated for analysis and application. These seven themes of data are cadastral, digital orthoimagery, elevation, geodetic control, governmental units, hydrography and transportation. The primary purpose of Framework is to develop procedures, technology, and standards that provide for integration, sharing, and use of the seven data themes.

The American National Standards Institute (ANSI) Geographic Information Framework Data Content Standard (FDCS) was developed by thematic experts and is based on conceptual models specified in international standards. It was developed with conceptual data modeling principles to ensure that the data content is comprehensive enough to satisfy a wide range of applications and flexible enough to handle future developments. The FDCS is designed to accommodate the development of interoperable applications that make data more accessible to users and easier to integrate with other sources. There are many advantages to implementing the Standard, including improved service and decision-making, less redundant data production and lower maintenance costs. Ready access to reliable, standardized data for decision making is important in many business cases and can be critical in emergency response situations. Other sectors are also realizing the importance of data standards, including commerce, real estate, engineering and facilities management.

One of the major issues organizations face is the cost of converting their data to conform to standards. In addition, applications that use the data are often dependent on the existing data structure, which further increases the cost of change. This guidance document was developed to address these issues with the FDCS. Data producers have the option to harmonize their data to the FDCS, instead of actually having to change their data. Harmonization is accomplished by matching existing data entities to Framework entities, and using an automated tool to transform the data into a standardized content. In this document, complex data model diagrams from the Standard are broken down into components in order to simplify the harmonization (or matching) process. As a result, data can remain in its original structure and format for internal organizational requirements. At the same time the standardized content can be made available via interoperable applications for a variety of other uses.

Used as a reference along with the FDCS publication, this document will facilitate the process of creating new standardized data, harmonizing and transforming existing data to match standardized content, and programming applications that integrate data from different sources. Background information is provided on data modeling methodology and terms, in order to clarify the base requirements for how to describe data content. Data may be structured in very different ways (even within the same theme), but a standard way of describing the data content makes it much easier to develop applications to facilitate data acquisition and use.

# 1. Introduction

## 1.1 Purpose of this document and intended audience

The purpose of this Guidance Document is to provide a reference to the Geographic Information Framework Data Content Standard, produced by the Federal Geographic Data Committee (FGDC) as part of the National Spatial Data Infrastructure (NSDI). The Guidance Document provides background information about the requirements of the Standard, and detailed explanations of the data models for each of the seven NSDI Framework themes.

The intended audience is implementers of the Geographic Information Framework Data Content Standard (hereafter referred to as the Framework Standard). This may include geospatial data producers, data managers, database designers, database programmers and application programmers. Managers at all levels may find useful background information and motivation for implementing Framework in the Executive Summary and the Introduction.

Used along with the published Framework Standard, this document will greatly assist the process of developing data that is compliant with the Framework Standard. Compliance can be accomplished several different ways: by creating new data, transforming existing data, or by programming tools which transform data at will, allowing data to stay in its original structure for specific organizational needs. The Framework Data Content Standard will also facilitate the programming of applications that integrate data from different sources.

This document also provides references on current and planned implementation methods, including tools for transforming data from one model to another, validating data models for compliance, and tools for accessing and sharing Framework data (e.g. Framework data services).

This document should be used in conjunction with the Framework Standard Base Document (Part 0), in addition to specific thematic parts of the standard (Parts 1 through 7) of interest to particular users. All the parts of the Framework Standard may be found at *http://www.fgdc.gov/standards/standards_publications/* under the section titled “Geographic Information Framework Data Standard.”

For clarification, the term “sections” is used to refer to specific sections of Parts 0 through 7 of the Framework Standard, and the terms “chapters” and “subchapters” are used to refer to specific sections of this guidance document.

## 1.2 How to use this document and related resources to start implementing Framework

This document is not designed to be read in entirety from front to back; it should be used as a reference. This subchapter provides a potential workflow for developing data compliant with the NSDI Framework Data Content Standard and making it accessible for sharing and integration with other data sources. The workflow includes references to applicable subchapters in this guidance document, sections of the Framework Standard, or other resources.

1. Determine if your data falls into one of the seven thematic categories of data covered by the Framework Data Content Standard (see subchapter 1.3.1)

2. Understand the need for the Framework Data Content Standard and benefits of using it.

    a. This data content standard structures data and information about data such that computers can communicate between each other to determine whether or not data sets are compatible for a particular application, or integration with other data.

    b. Using open standards such as the Framework Standard has been shown to reduce risk and increase return on investment. These conclusions are based on a "Geospatial Interoperability Return on Investment Study" done for NASA in 2005 (http://www.egy.org/files/ROI_Study.pdf) and other case studies described in the Spatial Data Infrastructure Cookbook (http://www.gsdi.org/gsdicookbookindex). For more benefits of NSDI Framework see subchapter 1.4 and subchapter 1.5.1.

    c. The Framework Data Content Standard does not modify an organization's internal business processes or how the organization creates and manages its data. Organizations that already conform to or plan to conform to other standards (for instance, state or industry standards) can also choose to implement the national Framework Data Content Standard specifically for data sharing. For more information on how Framework can provide a "middle ground" for data sharing, see subchapter 1.4.1.

3. Decide on an implementation strategy for conforming to the Framework Data Content Standard. The following choices are possible implementation strategies:

    a. Transform existing data content into content compliant with the Framework Standard.

    b. Create a transformation utility that converts existing data (on-the-fly or according to a regular schedule) into content compliant with the Framework Standard for sharing. Data stays in its original structure for internal business needs but is simultaneously available for sharing in a standard format.

    c. Create data content from scratch using the Framework Standard.

4. Become familiar with the thematic part(s) of the Framework Standard that correspond your data.

    a. Read the subchapter(s) of Chapter 3 of this guidance document that correspond to the thematic part of the Framework applicable to your data. For instance, if you have data on governmental units or administrative units, read subchapter 3.5.

    b. The simplified versions of the data models (shown as UML class diagrams) in these subchapters will provide an overview of the main features required for the thematic data, as well as their attributes. Required attributes are discussed, usually with examples, following the data model figures. An overview of optional features, relationships and attributes are also included.

5. If your implementation strategy is either 3a or 3b (transforming existing content to Framework Standard content), create a "matching" or "mapping" from the existing schema (features, attributes and relationships) to the Framework Standard schema. This process is called "schema matching" and is a required step for data harmonization/data transformation. Schema matching must be done manually by those most familiar with the data content and structure. Software is available to assist with schema matching; see subchapter 4.2 for resources.

    a. Use the data dictionary tables in the corresponding thematic part of the Framework Data Content Standard (parts 1 through 7) and find matches between features and attributes listed in the tables to corresponding features and attributes in existing data. For instance, if your data is governmental units, use Table 2, Data Dictionary for GeographicArea, and Table 3, Data Dictionary for GovernmentalUnit (from Part 5) to determine all the required content and find appropriate matches. See subchapter 2.3.2 for more information about data dictionary tables.

        i. All thematic parts of the Framework Standard require features which are defined by the Framework Feature Model (see subchapter 2.3.1.3) with corresponding permanent identifiers (see subchapter 2.3.1.5).

        ii. If your data has relationships (for instance, governmental units can share boundaries with other units) or associated attributes or events (for instance, transportation features such as roads may have speed limits associated with different lengths of road) you may need to reference additional data dictionary tables.

        iii. If your data has content that has no corresponding match in the Framework Standard, the ExtendedAttribute can be used as many times as needed to add additional content. See subchapter 2.3.1.3 for more information about the ExtendedAttribute.

        iv. If your data is lacking content required by the Framework Standard, you should still create the required features and attributes in your schema, even if you are unable to populate them. This should be noted in the metadata. See subchapter 2.3.3 for more information about required metadata.

        v. Schema matching may require some background knowledge on data modeling, particularly the importance of levels of abstraction and how it is implemented in schemas. For instance, to understand why both GovernementalUnits and AdministrativeUnits require attributes from an additional data dictionary table, GeographicArea, it is important to understand levels of abstraction and inheritance. For more information, see subchapter 2.2.

        vi. Schema matching may also require knowledge of UML terminology, which is described in subchapters 2.3.1.1 and 2.3.1.3. These subchapters are particularly helpful if the selected implementation strategy involves programming a transformation utility to convert data.

b. After matching attributes, use codelist and enumeration tables in the corresponding thematic part of the Framework Data Content Standard (parts 1 through 7) to find standard values and descriptions for specific attributes. See subchapter 2.3.1.3 for more information about codelists and enumerations. The use of standardized values will allow applications to recognize specific data for specific purposes and increase interoperability. For instance, the Governmental Units part of the Framework Standard (part 5) includes a codelist called "GovernmentalUnitType" which includes names and definitions for the most common instances of governmental units.

c. After schema matching is accomplished, data content in its original format can be transformed to the Framework Standard content. This can be either a one-time step, or can be performed "on-the-fly" if the schema matching is programmed as a rule-based transformation. Virtually any programming environment can be used to build such a transformation utility; see subchapter 4.2 for more resources.

6. If your implementation strategy is 3c (creating data from scratch to conform to the Framework Standard content), use the data dictionary tables in the corresponding thematic part of the Framework Data Content Standard (parts 1 through 7) to create a schema with all required features and attributes (e.g. mandatory elements). Content can be created that uses a relational database model, an object-oriented data model, a flat-file system or whatever type of structural model best fits your organization's data needs. See subchapter 4.2 for resources to assist with creating schema.

   a. All thematic parts of the Framework Standard require features which are defined by the Framework Feature Model (see subchapter 2.3.1.3) with corresponding permanent identifiers (see subchapter 2.3.1.5).

   b. Decide which optional elements from the data dictionary tables you would like to implement.

   c. Use codelists and enumeration tables in the corresponding thematic part of the Framework Data Content Standard (parts 1 through 7) to find standard values for specific attributes. See subchapter 2.3.1.3 for more information about codelists and enumerations.

   d. Create metadata and associate it with your data. See subchapter 2.3.3 for more information about required metadata.

7. Decide on an implementation strategy for sharing data. Both the Open Geospatial Consortium (OGC) and the FGDC recommend using a WFS (web feature service) to provide data in Geography Markup Language (GML), an open standard encoding. See subchapter 4.3 for resources concerning WFS and related services.

## 1.3 Overview of NSDI Framework

In 1994, Presidential Executive Order 12906 created the National Spatial Data Infrastructure (NSDI), defined as “the technology, policies, standards, and human resources necessary to acquire, process, store, distribute, and improve utilization of geospatial data.” One of *six basic building blocks of the NSDI*, Framework is a collaborative initiative to develop geographic datasets that are compatible based upon spatial location and content.

Framework data is a set of common themes of geospatial data that provide the basic data "skeleton" needed by users of Geographic Information Systems (GIS). The data layers that make up the Framework are the bones on which state agencies, local and county governments, tribal governments, academic GIS users, and the private sector can build their own GIS data. In many ways, the notion of framework data is analogous to the categories of information compiled for and portrayed on paper “base” maps, such as the United States Geological Survey’s (USGS) topographic map series. Different themes such as topography, hydrography, and transportation are represented with specific colors and symbols on a single map sheet upon which measurement and analysis may take place. The topographic map series’ organization of data at multiple map scales, using a nested tiling scheme, also corresponds with the seamless, integrated, multi-resolution nature of framework data.

Framework data layers are intended to be made available to the user community as freely and as easily as possible. Not only does Framework ensure trustworthy data – the best available data for an area, certified, and described according to a common standard, it also includes procedures, guidelines, and technology to enable participants to integrate, maintain, and distribute Framework data. Organizations can funnel their resources into applications, rather than duplicating data production efforts.

*Key aspects of the NSDI Framework:*

- Seven themes of most commonly used digital geospatial data
- Procedures, technology, and guidelines that provide for integration, sharing, and use of these data.
- Institutional relationships and business practices that encourage the maintenance and use of data.

*Key benefits of Framework:*

- Facilitate production and use of geographic data
- Reduce overall operating costs for geographic data clients
- Improve service and decision-making

**The ultimate goal of the Framework Data Content Standard is to provide uniformity to data and data sources. In doing so data becomes accessible to more users, and reduces cost in data redundancy.**

The need for detailed, up-to-date information for emergency response is one critical example of a practical need for Framework. In times of crisis, first responders of all disciplines rely on continuous streams of detailed and updated information to effectively respond to disasters and speed recovery. In particular, geospatial information facilitates informed decisions and improves communication throughout the emergency response community.

There are many other practical uses of Framework. A regional transportation planning project can use base data supplied by the localities it spans. Local, state, and federal agencies can respond quickly to a natural disaster by combining data. A county can use watershed data from beyond its boundaries to manage its water resources. State agencies can better track the ownership of publicly held lands by working with local governments' parcel data. State and local governments can more easily comply with federal reporting requirements.

### *1.3.1 Seven Framework data themes*

Geographic data users from many disciplines have a recurring need for a few themes of basic data. Identification of these framework themes resulted from numerous surveys administered in the mid 1990's by the US Geological Survey, the National Research Council and the National Center for Geographic Information and Analysis. The seven framework data themes identified are: cadastral information, digital orthoimagery, elevation, geodetic control, governmental units, hydrography and transportation. Many organizations produce and use such data every day. These themes comprise the core geospatial data used by most Geographic Information Systems (GIS) applications. Essentially, the NSDI seeks to assemble this basic geospatial data nationwide and the Framework is the foundation for this effort.

Cadastral information
Cadastral information describes the geographic extent of past, current, and future right, title, and interest in real property, including surface, above and below ground and water, and the foundation to support the description of that geographic extent. Cadastral information can include reference systems such as the Public Land Survey System (PLSS) and similar systems for areas not covered by the PLSS. It also includes publicly administered parcels, such as military reservations, national forests, and state parks, as well as marine jurisdictions. The Framework also provides a means to link existing parcel data into the larger cadastral network.

Digital Orthoimagery
Digital orthoimagery are georeferenced images of the Earth's surface, collected by a sensor in which image object displacement has been removed for sensor distortions and orientation and for terrain relief. Many geographic features, including those that are part of the Framework, can be interpreted and compiled from an orthoimage. Orthoimages can also serve as a backdrop to reference the results of an application to the landscape. The Framework may include imagery that varies in resolution from sub meter to tens of meters. Accurately positioned, high-resolution data (pixels of 1 meter or finer) are presumed to be the most useful for supporting the compilation of Framework features, particularly those that support local data needs.

Elevation
Elevation data provides information about terrain. Elevation refers to a spatially referenced vertical position above or below a datum surface. The Framework includes the elevations of land surfaces and the depths below water surfaces (bathymetry). Elevation and bathymetry may be modeled in various forms, such as in an evenly spaced grid or as irregularly spaced points (triangulated irregular network, hypsography, mass points). Elevation data is used to create representations of the terrain, such as a contour map, spot elevations, or a three-dimensional perspective view. Elevation data are also used to build models and perform applications, ranging from line-of-sight calculations, to road planning, to water runoff.

Geodetic Control
Geodetic control provides a common, consistent, and accurate reference system for establishing coordinates for all geographic data. The fundamental geodetic control for the United States is provided through the National Spatial Reference System (NSRS) managed by the National Oceanic and Atmospheric Administration (NOAA). Geodetic control information plays a crucial role in developing all Framework data and users' applications data, because it provides the spatial reference source to register all other spatial data. In addition, geodetic control information may be used to plan surveys, assess data quality, plan data collection and conversion, and fit new areas of data into existing coverages.

Governmental units
Governmental units include the nation, tribes, states, counties and statistically equivalent areas, incorporated places and consolidated cities, functioning and legal minor civil divisions, federal- or state-recognized American Indian reservations and trust lands, and Alaska Native regional corporations.

Governmental unit boundaries are used for a wide variety of applications. Some need the boundaries only for information and orientation; others require the polygons to determine inclusion related to a number of other features. Business geographics, which uses GIS to help businesses with planning and increasing sales, is a very active field that uses these boundaries for statistical analysis and decision making.

**Federal agency maintenance authority for each theme:**

Cadastral
- Bureau of Land Management Cadastral Survey

Digital Orthoimagery
- U.S. Geological Survey

Elevation
- U.S. Geological Survey

Geodetic Control
- National Geodetic Survey, NOAA

Governmental Units
- U.S. Census Bureau

Hydrography
- U.S. Geological Survey
- U.S. Environmental Protection Agency

Transportation
- U.S. Department of Transportation (USDOT)

Hydrography
Hydrography includes surface water features such as lakes, ponds, streams and rivers, canals, oceans, and coastlines. Each hydrography feature is assigned a permanent feature identification code and may also be identified by a feature name. Spatial positions of features are encoded as flowlines and polygons. Network connectivity, direction of flow, and a linear referencing system are also encoded.
Hydrography is important to many applications. As with other data themes, many users need hydrographic features as reference or base map data. Other applications, particularly environmentally oriented analyses, need the information for analysis and modeling of water supply, pollution, flood hazard, wildlife, development, and land suitability.

Transportation
Transportation data are used to model the geographic locations, interconnectedness, and characteristics of the transportation system within the United States. The transportation system includes both physical and non-physical components representing all modes of travel that allow the movement of goods and people between locations. Sub-themes representing the physical components of the transportation infrastructure include the road, railroad, transit, and waterway networks, plus airport facilities. Transportation features and related data are important elements of many planning applications. Geocoding applications use road and related address data for uses ranging from marketing analysis to

site identification. Routing applications use street network data for operations such as vehicle dispatch and fleet management.

### *1.3.2 Framework data organizations*

The Federal Geographic Data Committee (FGDC), Geospatial One-Stop (GOS), and The National Map are three national geospatial initiatives that share the goal of building the NSDI. FGDC focuses on policy, standards, and advocacy; GOS focuses on data discovery and access; and The National Map focuses on integrated, certified base mapping content. The National Geospatial Programs Office (NGPO) of the U.S. Geological Survey is the organizational host for these complementary activities.

**Partnerships are Key**

Similar government entities, such as neighboring counties, or a county and city within its boundaries, may partner to combine data for a larger area. Two states may collaborate for an issue that crosses their borders. Two federal agencies may have a common interest in an area. Public and private sector organizations, such as utilities and local governments, often have common land interests.

While the above entities are key players in providing guidance and tools to implement Framework, the data itself comes from multiple sources: local, state and federal government, tribes, and the private sector. The Framework is built from the ground up as organizations share data and coordinate data development.

Challenges to the geospatial community are often not as much related to technology as they are in managing the quantity and quality of available information, synthesizing that information to ensure completeness and consistency, making this information readily available and interoperable, and – particularly – developing business models that stimulate data sharing and commitments to long-term information stewardship.

In recognition of the above-mentioned challenges, the FGDC/NSDI Future Directions Initiative and the National States Geospatial Information Council (NSGIC) developed the *Fifty States Initiative Action Plan* in 2005. The goal is to put in place effective statewide coordination mechanisms in order to leverage the day-to-day efforts of all levels of government (tribal, state, regional and local government) towards building the NSDI.  In addition, a comprehensive *NSDI training program* covers a broad range of topics from geospatial metadata, geospatial web services, and standards, including the Framework Standard.

## 1.4 Overview of Framework Data Content Standard

The Framework Standard specifies a minimal level of data content and requirements that data producers, consumers, and vendors are expected to use for the interchange of the seven Framework data themes.  The data standard allows computer software to use a set of characteristics to parse data sets and test for compatibility.

- The standard does not specify a specific computer file structure. Specific computerized implementations of the standard may be developed (for instance an implementation of the standard has been developed using Geography Markup Language).

- It does not define how a user must create data, but it does define what elements should be included when data is created, and how these elements should be related, in order to accomplish successful interchange and integration of data.
- The Framework Standard is not a transfer standard; rather it is intended to structure data and information about data such that computers can communicate between each other to determine whether or not data sets are compatible for a certain project.

### 1.4.1 Need for standards

Standards enable data sharing and data integration. This includes exchange between data producers and users, and also between different computer systems (interoperability). Framework leverages individual geospatial data efforts so data can be shared. Implementing Framework data standards means that geospatial data shares common semantics, which makes it more accessible.

Data standards will help address these common problems:

- Most organizations need more data than they can afford. Often, large amounts of money are spent on basic geospatial data, leaving a shortage of funds for applications and project development. Some organizations cannot afford to collect base information at all.
- Organizations often need data outside their jurisdictions or operational areas. They do not collect these data themselves, but other organizations do.
- Data collected by different organizations are difficult to integrate, often due to differences in semantics. Therefore, information needed to solve cross-jurisdictional problems is not readily available.
- Duplication of data and applications results from lack of access to existing data, or inability to integrate data.

The need for data standards is not limited to the field of geographic information. The importance of standards is recognized in areas as diverse as engineering, commerce, real estate and facility management. The following statements regarding real estate and facilities management show this area's need for standards, and also highlight the parallels to needs with geographic information:

> Web-based technology has allowed facility executives to keep their fingers on the pulse of the organization as never before. But the technology also points to the need for solid data management procedures because a Web-based management system is ultimately only as good as the data that is fed into it. "The technology has made leaps and bounds," Fuhrman says [Andy Furman, executive director of OSCRE, the Open Standards Consortium for Real Estate]. "Now the data needs to catch up."
>
> Making that change will require significant advances across the industry. For some companies, facilities-related data is so fragmented that it requires the services of a consultant to put the information in order…. A common barrier to interoperable data is that different business units are often not communicating to each other, which leads to work being duplicated (Lorenz, 2006).

In addition to the same problem with duplication of data that the NSDI seeks to address, the real estate and facilities management sector also recognizes the need for common semantics, which simply means people using the same terms to identify the same objects or processes. Common semantics is referred to as standardized terms or business terminology in the following statements:

> It can be difficult to sell a CEO on the need to improve data interoperability within an organization… because it is difficult to measure the return on investment. But organizations are beginning to see the importance of having standardized terms, according to Fuhrman. “The building owners, investors and operators are becoming more aware of the need for standardization,” he says. “The next big leap for the industry is going to be the agreement on business terminology. That is where you are going to see a return on investment.”
>
> In addition to creating savings in the facility management phase of the life cycle, data interoperability offers the ability to speed business decisions. “If it takes you six months to do a space audit, the opportunity to make a quick business decision has been lost. These days, if you can’t produce relevant data with which you can make strategic decisions within a week, you’ve really failed in your role as a facility executive” (Lorenz, 2006).

Just as in the case of facilities management described above, the accessibility of geographic information can also be tied, in some cases, to the need for quick business decisions and most certainly in the case of emergency response.

Data content standards provide GIS data developers with common semantics and also with a template to use and expand, as needed, rather than developing their own from scratch. As more organizations participate, the data pool widens, and more resource savings and operational benefits are realized. In addition, Framework data can be used to address phenomena that do not adhere to jurisdictional boundaries or predefined regions. Users can perform cross-jurisdictional and cross-organizational analyses and operations.

While the benefits of standards are readily seen, one of the major issues organizations face is the cost of converting their data to conform to standards. In addition, applications that use the data are often dependant on the existing data structure, which increases the cost of change. The development of the NSDI Framework Data Content Standard takes these issues into consideration. The Standard does not require a particular structure for the interchange of data. Further, data producers and users may structure framework data in any way for their own internal use. The standard does not modify an organization’s internal business processes or how the organization holds data, except for the requirement that a minimal level of data content is present.

Instead of being required to change their data to match the Framework Data Content Standard, data producers can harmonize their data to the Framework Standard. This is accomplished through matching existing data entities to Framework. Figure 1.1 shows an example of how a schema, e.g. structure, of a database of roads can be harmonized to the roads part of the Framework Standard (Part 7c).

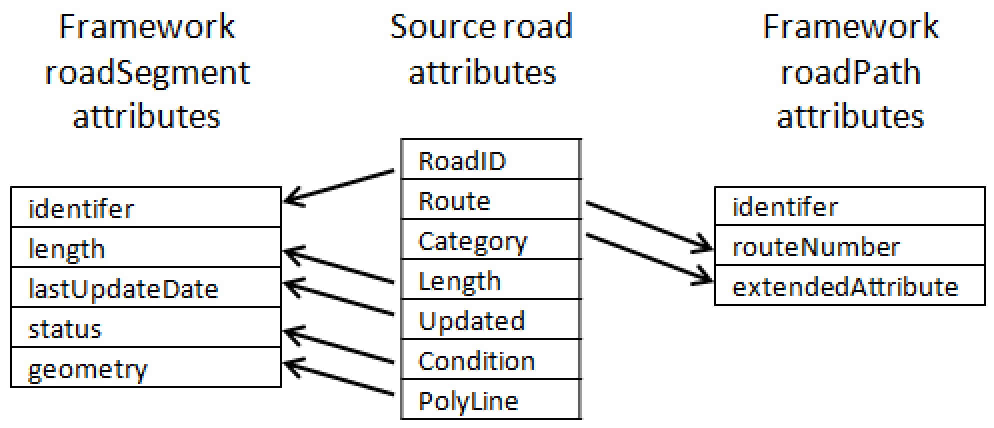


**Figure 1.1**
**Harmonizing one source of road data to the Framework road structure. Though the source data is only one entity and the matching Framework structure involves two entities, all the information from the source database can be harmonized to the Framework Standard.**

Based on this matching, tools can be programmed to transform data at will. This means organizations can retain their data in its original structure and format for their own applications, while at the same time sharing it via web-based services in a standard-compliant and interoperable form.

Additional benefits to adoption of the Framework Data Content Standard are discussed in subchapter 1.5.1.

### 1.4.2 Development of standards

The ISO/TC211 (International Organization for Standardization, Technical Committee 211 for Geographic Information) and OGC (Open Geospatial Consortium) have been actively developing international GIS standards that serve as a common and consensus framework for geospatial data and services. By successfully integrating the requirements for geographic information and innovative information technology, these standards have been widely adopted by many countries or regions, such as the United States, Canada, Japan, the European Union, Australia, and New Zealand, as the fundamental framework for developing their national spatial data infrastructures.

In order to insure that the Framework Data Content Standard meets the basic needs of all sectors in the United States, the Standard was developed through the American National Standards Institute (*ANSI*). ANSI accredits the InterNational Committee for Information Technology Standards (*INCITS*) as a standards development organization. Therefore, the Framework Data Standard development was able to take advantage of both ISO and ANSI standards expertise provided by members of INCITS Technical Committee L1, Geographic Information Systems.

The Standard was initially developed, in 2002, through the Geospatial One-Stop e-government initiative. Each standard part, relating to each of the seven Framework themes, was developed and edited by thematic experts. Responsibility for further development was transferred to the FGDC in 2003. The FGDC conducted a public review of the framework data standard in 2004, with nearly 5000 resulting comments. In 2006, further review of the draft standard and comments were undertaken by members of INCITS L1. ANSI submitted a notice for public review in early 2007. The standard was finalized, and then published in May 2008.

The Framework data standard is a voluntary consensus standard. *OMB Circular A-119* directs Federal agencies to use voluntary consensus standards, such as ANSI standards, however there is no legislation at the federal or state level that requires the use of this standard, nor is there a formal certification process. See Chapter 4, Implementing Framework Standard, for a discussion of various implementation projects including development of software for validation or conformance to the Framework Data Standard.

Many other nations have developed or are in the process of developing framework data and data standards. The Infrastructure for Spatial Information in the European Community (INSPIRE) directive is legislation to which all European Union member nations are bound, with prime objectives being data exchange and data sharing for effective governance and policy-making. Similar to the NSDI, the INSPIRE directive includes standardization of data so that it can be seamlessly combined across borders. While INSPIRE identified thirty-four spatial data themes (considerably more than the seven NSDI framework themes), data identified as the first order of priority for standardization includes five of the same themes identified by the NSDI framework. Taiwan's National Geographic Information System considers standardization as one of the key issues to successfully facilitate the sharing of geospatial data, with 14 core and sub-core data standards proposed to serve as the fundamental framework for geospatial data applications. Along a similar track, the Australian Spatial Data Infrastructure has objectives in place to support and promulgate a minimum set of "best practice" data standards (primarily based on the same ISO standards used by NSDI Framework) that facilitate data integration.

## 1.5 Structure of the Framework Data Standard

The Framework Data Content Standards are a suite of standard parts that collectively define minimum requirements for data to be considered Framework data. The Base Document (part 0) provides an overview and background information necessary for understanding each of the seven Thematic Framework Standards (parts 1-7). Each of the seven themes has its own standard part which specifies a minimal level of data content and structure in order to allow for interchange of data. The Transportation theme has its own Base part (part 7) and five additional parts (7a, 7b, 7c, 7d and 7e) to address different sub-themes such as roads, rails, transit etc.

All parts of the Framework Standard (parts 0 through 7) follow a similar structure, with sections that address each of the following topics:

- Introduction
- Scope, Purpose and Application
- Conformance
- Normative references
- Maintenance authority
- Terms and definitions, symbols and other notations
- Requirements
- Annexes

An overview of each of these subchapters follows to familiarize new users with how the parts of the Framework Standard are organized. In each part, many of these sections have the exact same wording or very similar wording. As each of the individual parts of the Framework Standard are discussed in more detail in Chapter 2 (the base part) and Chapter 3 (the thematic parts), the focus here is on the unique aspects of each part, and statements that are the same or very similar for each of the parts are not discussed.

### 1.5.1 Introduction

The Introduction section of all the parts of the Framework Standard begins with a declaration of the purpose of the Standard, such as the statement below, or a similarly worded statement:

> The primary purpose of this part of the Geographic Information Framework Data Content Standard is to support the exchange of NSDI framework [insert name of specific theme] data.

The Introduction in each part also includes a short summary of the benefits associated with use of the Framework Standard. Most parts utilize the statement below, or a similarly worded statement:

> This part seeks to establish a common baseline for the semantic content that will ensure the widest utility of thematic data for the user and producer communities. It also seeks to decrease the costs and simplify the exchange of governmental unit and other geographic area boundary data among local, Tribal, State, and Federal users and producers. That, in turn, discourages duplicative data collection. Benefits of adopting this part of the standard also include the long-term improvement of the thematic data within the community.

<u>Common baseline for semantic content:</u>
The goal behind common semantics is every user of a particular theme of geographic information, such as cadastral data, will understand exactly what is meant by a term, such as “parcel identifier”, and, when sharing data, be willing to adopt the use of this term instead of using other related terms. The philosophy of the Framework Standard is not to provide an exhaustive list of standardized terms for every possible instance, but to focus on standardizing terms that are commonly used by different parts of the community, both creators and users of Framework data. By focusing on key elements instead of exhaustive lists, the goal is to make the Framework data useful to the widest range of users and producers of this data.

<u>Decrease costs and discourage duplicative data collection:</u>
As an example of implementation of the Framework Data Content Standard and sharing of data via interoperable web-services, a local government will be able to continuously maintain and update its land records while serving them into other parts of the organization as well as to external organizations. A utility company could then directly use the base map of the local government as a replacement for its own base maps for facility data, resulting in a significant savings of cost of having to maintain and update a dataset that is essentially a duplicate. The utility company could also serve its facilities data back to a local government for use in permitting and land use planning.

<u>Long-term improvement of the thematic data within the community:</u>
If geographic framework data is made available via interoperable web-services, increased access to the data from different organizations and users in the community can help pinpoint areas where data is missing, overlapping, contains errors, or needs updating. Different organizations can also identify shared

data needs and pool resources for development of new data or updates to increase the accuracy, extent, or content of existing data.

### 1.5.2 Scope, Purpose, Application

All parts of the Framework Standard include a section on Scope, which is a description of the range covered by the part of the Standard including what parts it does not cover. For instance, Part 1: Cadastral "provides the information necessary to identify the existence of parcel-level cadastral information and the source of that information" and it also specifically states that "this part of the standard is **not intended** to support homeland security, citizen query and access, real estate records, or other application-based information."

Most of the parts of the Framework Standard contain similar wording pertaining to Scope, provided to underline one of the key benefits of the Framework Standard: interoperability. Here are several examples:

- Part 6: Hydrography states that "This part specifies the content and its organization necessary for the successful interchange of hydrography data. This part does not specify a particular structure for the storage of hydrography data."
- Part 5: Governmental Units states "This part does not specify a particular structure for interchange of boundary data. Further, data producers and users may structure geographic area boundary data in any format for their internal use."
- Part 7a: Transportation states "Transportation data can be implemented using a variety of software packages. It supports the mapping and conversion of native data in any format into a common representation for exchange over the Web or as files."

In addition to Scope, some parts of the Framework Standard also include a Purpose section. The Purpose section restates the primary purpose as stated in the Introduction: "to support the exchange of data" and may elaborate on how this achieved. For instance, most parts also state "It is the intent of this part to set a common baseline of information content for exchange within the thematic data community that will enhance data sharing and applications development when used with standards-based Web services or file transfer." Other purposes, specific to the theme addressed, may be included in this section.

Some parts of the Framework Standard also include an Application section (or in the case of Part 6: Hydrography, a Capabilities section). This section contains a description of applications specific to particular themes. A few examples from different thematic parts are provided below:

- Part 1: Cadastral states "The Cadastral part profile supports the discovery of and the navigation to cadastral information. Associated metadata will identify the providers of additional cadastral information."
- Part 4: Geodetic Control states "this part is applicable to any geodetic control dataset and is intended to facilitate a common methodology to create, manage, and share geodetic control datasets from various organizations…. Although this part does not encompass non-geodetic control points … it can be used as a model for other control points and coordinated points."

### 1.5.3 Conformance

To conform to the Framework Standard, a thematic dataset must:

1) **meet the requirements of the data dictionary** for that theme as described in the thematic part of the standard.
2) **be accompanied by metadata** which conform to at least the minimal set of mandatory elements of either ISO 19115, Geographic Information. Metadata, or FGDC-STD-001-1998, Content Standard for Digital Geospatial Metadata.

The Base document of the Standard contains a large volume of information to define and express data content and relationships, including references to other standards. All of this information may lead to this misconception there are additional requirements for conformance other than the two described above.

Conformance to the standard does not mean the data has to be in a specific format, or even a specific database structure (such as a relational database structure, or object-oriented structure). Conformance means that a minimal level of data content is present and is described according to the terms, definitions and domains provided by the Framework Standard. Additional content may be included as optional elements.

Each thematic part contains the following statement pertaining to conformance:

> Each thematic part of the Framework Data Content Standard includes a data dictionary based on the conceptual schema (i.e. data model) presented in that part. To conform to this standard, a thematic dataset shall satisfy the requirements of the data dictionary for that theme. It shall include a value for each mandatory element, and a value for each conditional element for which the condition is true. It may contain values for any optional element. The data type of each value shall be that specified for the element in the data dictionary and the value shall lie within the domain specified for the element.

Data dictionaries and their requirements are discussed in more detail in subchapter 2.3.3.

Without comparing a data dictionary for a particular dataset to the data dictionary requirements specified in a thematic part of the Framework Standard, how can you know that data conforms to the Standard? "Framework data" is a not a definitive label. Many states have their own definition of framework data, which may or may not conform to the NSDI Framework Data Content Standard. Annex C of the Base document of the Standard describes a series of XML Schema Document files (.xsd) that can be used to validate geographic information for each framework data theme. XML editing software can be used to compare the schema (e.g. structure) of a particular database with the XML Schema Document files for the Framework Data Content Standard.

### 1.5.4 Normative references

All parts of the Framework Standard include a section about normative references, which contain information that is minimally required by the Framework Standard. Annex A of the Base Document (Part 0) lists normative references applicable to two or more parts, and any normative references that are specific to just one of the thematic parts of the standard are listed in the Normative References section of the thematic part.

Most of the parts of the Framework Standard also contain informative references which may contain helpful information or examples of implementation (but are not required). Annex D of the Base Document lists informative references applicable to two or more of the parts. Informative references specific to just one thematic part are included as an annex in that part's document.

### 1.5.5 Maintenance authority

Who is responsible for maintaining the various parts of the Framework Standard? The wording for this section in all parts of the Framework Standard is very similar. For all parts, the FGDC is the responsible organization for coordinating work on all parts of the Framework Data Content Standard.

In addition to the FGDC, another agency (or agencies) may be listed as responsible for working with the FGDC to maintain a particular thematic part. For instance:

> The U.S. Census Bureau on behalf of the U.S. Department of Commerce, working with the FGDC, is directly responsible for development and maintenance of the Geographic Information Framework Data Content Standard, Part 5: Governmental Unit and Other Geographic Area Boundaries.

The section also contains contact information for the FGDC and in most parts it also includes contact information for the other responsible agency.

### 1.5.6 Terms and definitions, symbols and other notations

All parts of the Framework Standard provide a section with an alphabetical listing of important terms, along with their definitions. The Base document provides a list of general terms and definitions that are used in at least two or more parts. Terms that are specific to just one part are listed only in that part's document.

Following the Terms and definitions section, every part of the Framework Standard includes a section listing abbreviations, symbols and other notations.

### 1.5.7 Requirements

All parts of the Framework Standard include this section, which defines the minimum data content required for the thematic parts (for Part 5: Governmental Units, this section is called Content model instead of Requirements).

The Requirements section of Part 0: Base is somewhat different, since the Base part does not refer to a particular theme of data. This section describes the requirements for how the data content in the other parts are described. For instance, data from all thematic parts must be described with an application schema (a data model) expressed as a UML class diagram accompanied by a data dictionary table.

### 1.5.8 Annexes

All parts of the Framework Standard include annexes, which can be of two kinds: normative and informative. Normative annexes include information that is required. Informative annexes include information that is not required but that may be helpful, including data examples.

# 2. Framework Data Content Standard - Base Document

## 2.1 Purpose

The purpose of the Base Document is a reference document to each of the seven thematic Framework Standard parts. The thematic parts define and describe conceptual data structures and relationships pertinent to each particular theme, using the methods outlined in the Base Document.

The Base Document provides a common foundation from which to implement the Framework Standards for each of the seven themes, resulting in more efficient implementation. The Base Document provides system architects, database designers, and software developers with methods for understanding the data content of the different themes so they can develop common methods of data exchange. Different sources of data may be structured in very different ways (even within the same theme), but a standard way of describing the data content makes it much easier to develop tools to facilitate data acquisition and use. Since all seven data themes are described using the same methods, it makes it easier for software developers to write software for parsing data based on its characteristics.

## 2.2 Background on data modeling

This section provides definitions and explanations of terms that are used in other sections of this guidance document:

- *conceptual data model*
- *logical data model*
- *physical data model*
- *universe of discourse (domain)*
- *abstraction*
- *instances*
- *entities (types)*
- *feature type*
- *generalization*
- *inheritance*
- *conceptual schema*
- *application schema*

The scope of the Base Document is to provide a high-level view of the seven framework data themes and an overall integrating model. This data model is expanded into greater detail in each of the thematic parts of the Standard (parts 1-7).

Data modeling methodologies are used to ensure that data can be managed and analyzed in a standard, consistent and predictable manner. Data modeling generally follows three stages: the conceptual data model, the logical data model, and the physical data model (Simsion, 2005).

The ***conceptual data model***, the focus of communication between business stakeholders, defines the semantics of data including data entities and their relationships. The ***logical data model*** is a translation of the conceptual model into structures for specific applications. The ***physical data model*** is how the logical data model is implemented and optimized in a particular database management system.

The conceptual modeling stage is important in order to produce a data model that is stable (meeting requirements) and yet flexible to change. If a data model is based on specific requirements without paying attention to future changes or integration with other data, this tends to make it inflexible.

A conceptual data model defines the concepts within a universe of discourse. The ***universe of discourse*** (also referred to as a *domain of discourse* or simply *domain*) is a view of the real or hypothetical world that includes everything of interest. Framework is part of the domain of users of geographic information, and is specific to the sharing of the most commonly required themes of geographic information in digital format. Its universe of discourse includes everything pertinent to the seven themes that make up Framework, but it also extends beyond these themes, to incorporate more abstract concepts from the broader domain of geographic information.

Because a conceptual model represents the semantics of a universe of discourse it uses various levels of ***abstraction***. Abstraction reduces and factors out details so that one can focus on a few concepts at a time; it is a way of handling complexity. The abstract perspective identifies the fundamental things or ***entities** (types)*, of which the actual data exemplifies or is an ***instance*** of (ter Bekke 1992).

One of the most important abstractions pertaining to Framework data is an entity called a ***feature type***, or simply *feature*. Features are abstractions of real-world phenomena or man-made constructs that typically have a persistent or assigned identity, such as a name or code, a location represented by a formalized geometry, and a set of other properties and relationships. Actual instances of this abstract concept are roads, streams, governmental units, and so forth. While many properties of a road differ from those of a stream, they share at least two things in common: a persistent or assigned identity (name or code) and a location. Roads and streams are types of features. Therefore a feature is an abstract entity that is common to all thematic parts of the Framework Data Content Standard (elevation and orthoimagery are classified as a special type of feature called a coverage, described in more detail in chapter 3). Identifying similar entities at a conceptual and abstract level helps to ensure the stability and flexibility of the Framework data model.

One of the most important concepts related to abstraction is ***generalization*** (ter Bekke 1992). Generalization is the recognition of similar properties of various types and combining these. For instance, a city has a boundary and a government. A county also has a boundary and a government. These entities share some properties; therefore they can be generalized into a new type, “governmental unit”. Furthermore, a governmental unit also has properties of identity (e.g. name) and location, so a governmental unit is a subtype of a feature type. Generalization is related to the idea of ***inheritance*.** If a governmental unit is a subtype of feature, then it inherits the properties of a feature. Similarly, since a county is a subtype of governmental unit, it inherits the properties of both feature and governmental unit.

A formal specification of a conceptual data model is called a ***conceptual schema***, and within the domain of geographic information, conceptual schemas are developed by the standards organizations ISO and OGC. As discussed in subchapter 1.4.2, ISO and OGC worked together to develop a conceptual schema (called an *abstract specification* by OGC) for the feature type. This conceptual schema is called the General Feature Model and is fully described in ISO 19109. The importance of ISO 19109 is summarized by the *Developing the Spatial Data Infrastructure: the SDI Cookbook*:

> Before anyone can allow software to reliably access mapped features stored in remote data systems, there must first be a common understanding about the nature and composition of the objects being managed. ISO 19109 includes guidance principles for classifying geographic objects. The usefulness of any information is reduced when the meaning is unclear, especially and commonly across different application domains. If different classifications are defined using a consistent set of rules, that ability to map one classification to another and retain the meaning will e greatly increased. This is also known as the semantic translation of one representation of an object in one system, for example a road or river segment, to that in another (Nebert 2004).

The model diagrams within the Framework Data Content Standard represent logical data models. More specifically, the diagrams provide information about the *application schema*[1], which is an implementation of formal conceptual schema(s) for a specific application. The conceptual schema for a feature may be used by many different types of applications. Framework may be considered one such application, with the purpose of sharing data. The base Framework application schema is designed to encompass the application of Framework as a whole – the sharing of a collection of data. This base Framework application schema is visualized as a UML class diagram in Figure 3 in Part 0: Base document and further described in a data dictionary table (Table 2 in Part 0). It is dependent on (in other words, inherits properties from) the conceptual schema for features in ISO 19109, General Feature Model.

More specific application schemas are provided for each of the seven thematic parts. Each of the thematic application schemas are dependent on (inherit the properties of) the Framework application schema, which in turn is dependent on the General Feature Model. In addition, some themes may also inherit properties of entities defined in other conceptual schemas. For instance, the abstract entity “coverage” is a specific type of feature used in elevation and orthoimagery themes. The coverage is defined in ISO 19123, Geographic information - Schema for coverage geometry and functions.

In summary, the Framework Data Content Standard is focused on the logical data model. These logical data models are created for general applications of Framework data, based on more abstract conceptual data models formalized in other standards. In turn, implementers of the Framework Data Content Standard can translate the logical data models into technology-specific physical data models. Implementers can apply the Standard to create a new data, in the physical data model of their choice that conforms to the Standard. In another scenario, an implementer could create an application that can access data from several different sources (different physical data models) for analysis.

## 2.3. Requirements

There are eight requirements listed in the Base Document. It is important to note that data does not need to adhere to these eight requirements in order to conform to the Framework Data Content Standard. As mentioned in subchapter 1.5.3, there are only *two* requirements for conformity (adherence to the data dictionary for a particular theme/subtheme, and metadata).

[1] The Base Document uses the ISO 19101 definition for application schema: “a conceptual schema for data required by one or more applications.”

The requirements listed in the Base Document are required methods of data content description. All thematic parts and subparts follow these requirements in the way that they provide information about the content of the data.

These are the eight requirements discussed in the Base Document (section 7):

7.1 Unified Modeling Language (UML) model
7.2 Dependence on ISO 19100 series of geographic information standards
7.3 Application schema
7.4 Data dictionary
7.5 Metadata
7.6 Model integration
7.7 Establishment of identifiers
7.8 Framework feature model and common classes

This guidance document divides these requirements into three subchapters: application schema (subchapter 2.3), data dictionary (subchapter 2.4) and metadata (subchapter 2.5). The rest of the requirements are addressed as subchapters of the application schema.

### 2.3.1 Application schema

An application schema is a formalized method of applying a conceptual schema to specific applications. An application schema is meant to increase productivity by maximizing compatibility and interoperability between systems, simplifying the process of design, and promoting communication between individuals and teams working on the system. A general methodology for the development of an application schema is provided in ISO 19109 Geographic information: Rules for application schema.

As stated in subchapter 2.2, the Base Document provides the base application schema for the Framework application as a whole. Each of the thematic Framework Data Content Standards includes more detailed application schema(s) which are dependent on the base Framework application schema. These application schemas specify, as appropriate, the following information:

- Feature types
- Attribute types
- Attribute domain
- Feature relationships
- Spatial representation
- Data organization
- Metadata

Framework requires that the application schemas be described with UML (Unified Modeling Language) class diagrams, accompanied by data dictionary tables. The application schemas also include definitions and sometimes figures and examples to clarify the semantics.

#### *2.3.1.1 Unified Modeling Language*

The subchapter includes definitions and explanations of the following terms:

- *UML class diagram*

- *rule-based transforms*
- *UML stereotype*

This subchapter refers to terms defined in other subchapters: *abstraction, generalization, conceptual data model, logical data model, physical data model (subchapter 2.2).*

The Unified Modeling Language (UML) is the method used to express the application schemas as diagrams. UML is an object modeling and specification language used in database development and software engineering. As a general purpose modeling language, it can be used to design, specify, visualize, construct, and document any sort of system.

A ***UML class diagram*** describes the structure of a system by defining the main entities (classes), their properties (attributes), and relationships with other classes. A UML class diagram is the best way to understand the generalization of different classes. For instance, certain combinations of lines and arrows connecting classes denote whether a class is a subtype of another more general class. The use of UML for data modeling allows the Framework Data Content Standard to be technology independent but permits current and future implementation cases to be derived from the UML model.

UML class diagrams can be used several ways: as a general conceptual model, a logical data model, or even a physical data model. In the Framework standard, UML class diagrams should be viewed as logical data models, instead of technology-specific physical data models, as stressed in section 7.1 of the Base Document:

> The use of UML class diagrams in the Framework Data Content Standard is an application neutral approach to depict the inherent description of and relationships among data entities. These diagrams should neither be interpreted as requiring object-oriented implementation – methods or interfaces are not typically shown on these data classes – nor should they be interpreted as representing tables in relational databases. Instead, the UML classes should be used as the basis for translation to and from internal organization data stores and applications. UML modeling environments typically support conversion of logical UML models into implementations in various programming environments through rule-based transforms.

Data models can be represented at various levels of generalization. For instance, the UML class diagrams in the Base Document are more generalized. UML class diagrams in the thematic parts are more detailed, providing specifics about subtypes of abstract entities. These more detailed UML class diagrams can be used to create physical data models, such relational tables or an object-oriented database. Using the UML class diagrams, programmers can transform data from two different sources (regardless of its physical structure) so that it can be integrated. ***Rule-based transforms*** are used to translate data into different formats tailored for a specific user or context. An example is translating certain elements from a database into a html-formatted table for web-based viewing. In the same sense, geospatial data created for use by a transportation department, for example, can be transformed to the Framework data model which enables it to be combined with other geospatial data for a need such as disaster response which involves the transportation department and many other agencies. By applying the rules expressed in a UML class diagram, data designed for one specific use can be transformed to work in an environment with other data and other uses it was not originally developed for.

It is important to note that programming is not required in order to transform data into compliance with the Framework Data Content Standard. Many databases provide means for altering schema, such as adding or changing attributes along with their data types and domains, outside of a programming environment. In addition, new data can be created to be Standard-compliant by importing the UML model into software for creating geospatial data.

For programmers who are interested in creating rule-based transforms, Annex B of the Base Document provides a description of UML notation as used in the Framework Data Content Standard. It is important to note that Framework has developed several UML stereotypes to extend UML concepts specifically for Framework data models. A ***UML stereotype*** extends the vocabulary of UML in order to create new model elements that have specific properties that are suitable for a specialized usage. Appendix B of the Base Document describes UML stereotypes specific to Framework, such as <<Feature>>, <<CodeList>>, <<Enumeration>> and <<DataType>>.

For the non-programmer, a basic description of UML notation is provided in the following subchapters so users can understand the UML diagrams that correspond to figures from the Base Document. For understanding of notations in UML diagrams in the thematic parts (chapter 3 of this guidance document), it is recommended that users refer back to subchapters 2.3.1.2, 2.3.1.3 and 2.3.1.4 or to Appendix B of the Base Document. Section 7.1 of the Base Document describes where the UML data models may be found in the thematic parts.

**FAQ:** Why does section 7.1 of the Base Document say that the UML data models in each thematic part are incorporated into the text in different places? Why isn't the UML found in the same place in each document?

Each thematic part of the standard is structured differently, in a way that best presents an understanding of the data content specific to that theme. Elevation data, for instance, may be represented in many different types of data models. Therefore the UML diagram for each data model is provided in the annexes of the Elevation standard (part 3).

#### *2.3.1.2 Dependence on ISO 19100 series of geographic information standards*

This subchapter includes definitions and explanations of the following terms:

- Classes
- Relationships
- Dependency

Section 7.2 of the Base Document states that all parts of the Framework Data Content Standard are dependent on concepts from these standards:

ISO 19107:2003, Geographic information - Spatial schema
ISO 19108:2002, Geographic information -Temporal schema
ISO 19109:2005, Geographic information - Rules for application schema
ISO 19111:2003, Geographic information - Spatial referencing by coordinates
ISO 19115:2003, Geographic information – Metadata

In addition, the digital orthoimagery and elevation data parts are dependent on ISO 19123:2005, Geographic information - Schema for coverage geometry and functions. Data standards for certain transportation modes are dependent on ISO 19133:2005, Geographic information - Location based services - Tracking and navigation.

**FAQ:** Do I need to purchase some or all of the above ISO standards in order to create or modify data so it is compliant to Framework?

No. Programmers may need access to some or all of these standards for implementation, but if you are creating or modifying data to the Standard through the use of database or GIS software, you will only need the appropriate thematic part of the Framework standard. Familiarity with one of the two standards metadata (FGDC or ISO) is also required.

Many of the UML class diagrams and data dictionaries throughout the parts of the Framework Standard will refer to classes prefixed with a two-letter code (e.g. GM_object, CV_Coverage) where the two-letter code refers to a model defined in one of the ISO standards. See Table 2.1 for a complete list of the prefixes used in all the Framework parts and the standard and model package they come from.

**Table 2.1**
**Prefixes identifying model packages from ISO standards that are used in the Framework Standard.**

| Prefix | International Standard | Model package |
|---|---|---|
| EX | ISO 19115 | Extent |
| DQ | ISO 19115 | Data Quality |
| GM | ISO 19107 | Geometry |
| TP | ISO 19107 | Topology |
| TM | ISO 19108 | Temporal Schema |
| GF | ISO 19109 | General Feature Model |
| SC | ISO 19111 | Spatial Coordinates |
| LR | ISO 19133 | Linear Reference Systems |
| CV | ISO 19123 | Coverages |

The UML diagram shown in Figure 2.1 shows, at the center, a rectangle labeled <<Application Schema>> Base. Rectangles are UML ***classes***, which represent the main entities (or objects, in UML terms), and the lines connecting them represent types of ***relationships*** between classes. The dashed lines mean that the type of relationship is a ***dependency*** and the class being pointed to by the arrow is required by the dependant class. The dependency relationship means that some of the classes in the Framework application schemas (represented as a whole by the class <<Application Schema>> in this diagram) require (will use) one or more classes from the conceptual schemas of ISO 19107, 19108, 19109 and so forth.

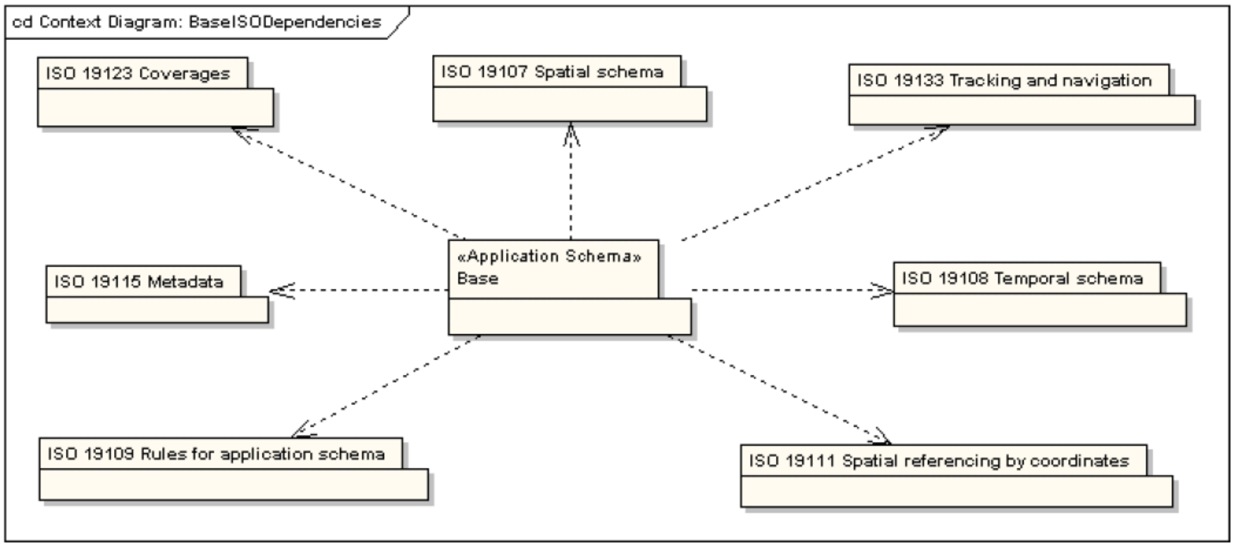

cd Context Diagram: BaseISODependencies
ISO 19123 Coverages
ISO 19107 Spatial schema
ISO 19133 Tracking and navigation
ISO 19115 Metadata
«Application Schema» Base
ISO 19108 Temporal schema
ISO 19109 Rules for application schema
ISO 19111 Spatial referencing by coordinates

**Figure 2.1**
**Framework base dependencies on ISO series of geographic information standards (reproduced from Base Document, Figure 1)**

#### *2.3.1.3 Framework feature model and common classes*

This subchapter includes definitions and explanations of the following terms:

- Attributes
- Data type
- Multiplicity
- Composition
- Role name

This subchapter refers to terms defined in other subchapters: *abstraction, instance, conceptual schema, application schema (subchapter 2.2), and UML classes and stereotypes (subchapter 2.3.1.2)*

The Framework Data Content Standard organizes information using the ISO General Feature Model found in ISO 19109. Features are abstractions of real-world phenomena or man-made constructs that typically have a persistent or assigned identity, such as a name or code, a location represented by a formalized geometry, and a set of other properties and relationships. The ISO General Feature Model provides a conceptual schema of features. This conceptual schema is used by Framework to construct a base application schema for Framework (Figure 2.2) and additional application schemas for specific themes, using rules for application schemas, also found in ISO 19109.

Each framework theme, represented by a part in the standard (parts 1-7), documents one or more formal feature types by creating instances of the Feature class shown in Figure 2.2. For instance, the Hydrography part has a <<Feature>> type called HydroElement which can be of several types: a segment of a stream or a canal, a part of a coastline, a body of water, etc. Because the attributes for a HydroElement will be similar regardless of type, these types are grouped under one feature. However, the Elevation part of the Standard has six different features types: ElevationGridCoverage, ElevationContourCoverage, ElevationTinCoverage, ElevationPointCoverage, ElevationPointSet and ElevationProfileCollection. Because these represent fundamentally different ways to represent and structure elevation data, they are treated as separate feature types.

Every thematic part in the standard also includes an instantiation of the FeatureCollection class. A FeatureCollection is an abstract representation of a data transfer. For instance, an ElevationCollection is particular instance of FeatureCollection that is used in Part 3: Elevation to represent a collection of elevation features for data transfer.

Figure 2.2 shows the relationship between the Feature class and the FeatureCollection class. UML notations used in this diagram are described in detail in the rest of this subchapter.

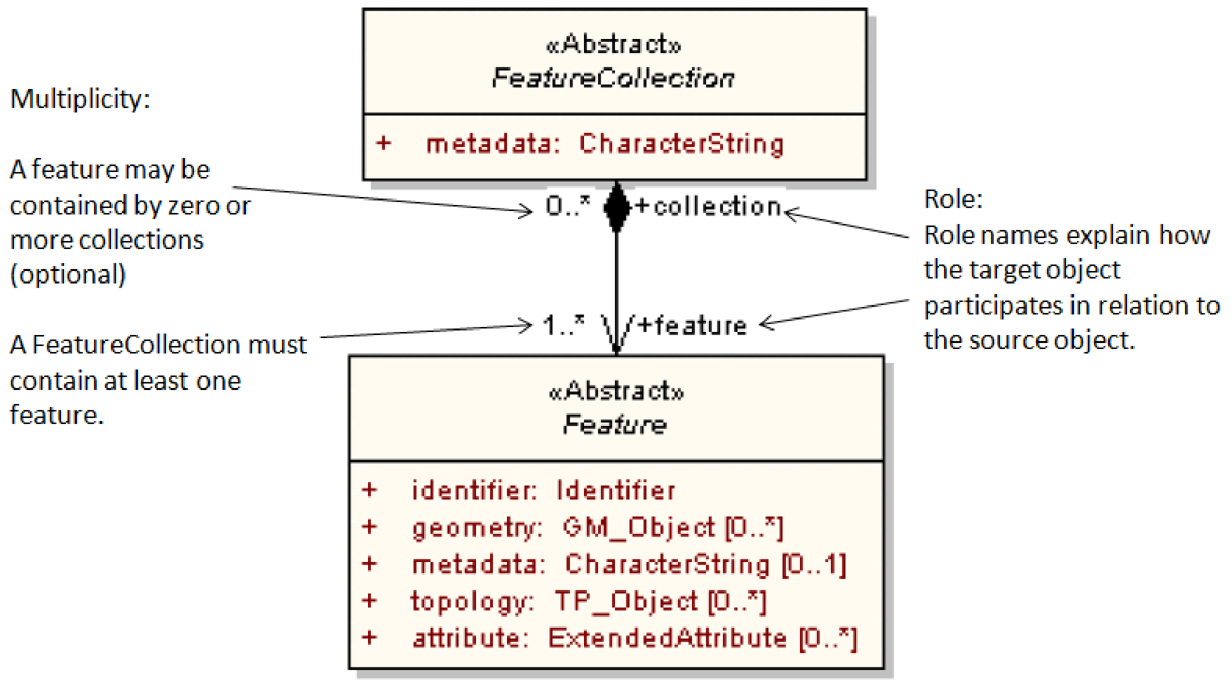

«Abstract»
FeatureCollection
+ metadata: CharacterString
Multiplicity:
A feature may be contained by zero or more collections (optional)
0..*
+collection
Role:
Role names explain how the target object participates in relation to the source object.
A FeatureCollection must contain at least one feature.
1..*
+feature
«Abstract»
Feature
+ identifier: Identifier
+ geometry: GM_Object [0..*]
+ metadata: CharacterString [0..1]
+ topology: TP_Object [0..*]
+ attribute: ExtendedAttribute [0..*]

**Figure 2.2**
**The two primary classes common to all parts of the Framework Data Content Standard, FeatureCollection and Feature (detail from Figure 3 in Part 0: Base document).**

There are several things to note about the UML notations in Figure 2.2. Compared to the classes in Figure 2.1, the classes shown here are rectangles that are divided and contain more information. The upper part of the class holds the name of the stereotype class specific to Framework in brackets.

The properties of the class are listed in the lower part of the rectangle, beneath the dividing line. In UML notation, properties are referred to as **attributes**.

Attributes have this structure: visibility name: type-expression – initial value [multiplicity]

- Visibility is public (plus sign "+") or private (minus sign "-")
- Name is a character string that is unique within the class

- Type-expression identifies the data type of the attribute (e.g. characterString, integer, boolean, dateTime). Additional data types may be defined as separate classes; see the following subchapter on <<DataType>> stereotype.
- Initial value specifies the default value for the attribute
- Multiplicity specifies the number of values that an instance of a class may have for a given attribute. If no multiplicity is given, that means exactly one instance is required. For instance the first attribute listed for <<Abstract>> feature class is identifier. Because no multiplicity is listed, it means that this is a mandatory attribute and it can occur only once – every instance of a feature must have a unique identifier. Some other examples are [0..*] zero or more instances (optional), [1..*] one of more instances (at least one), [2] exactly two instances and so forth.

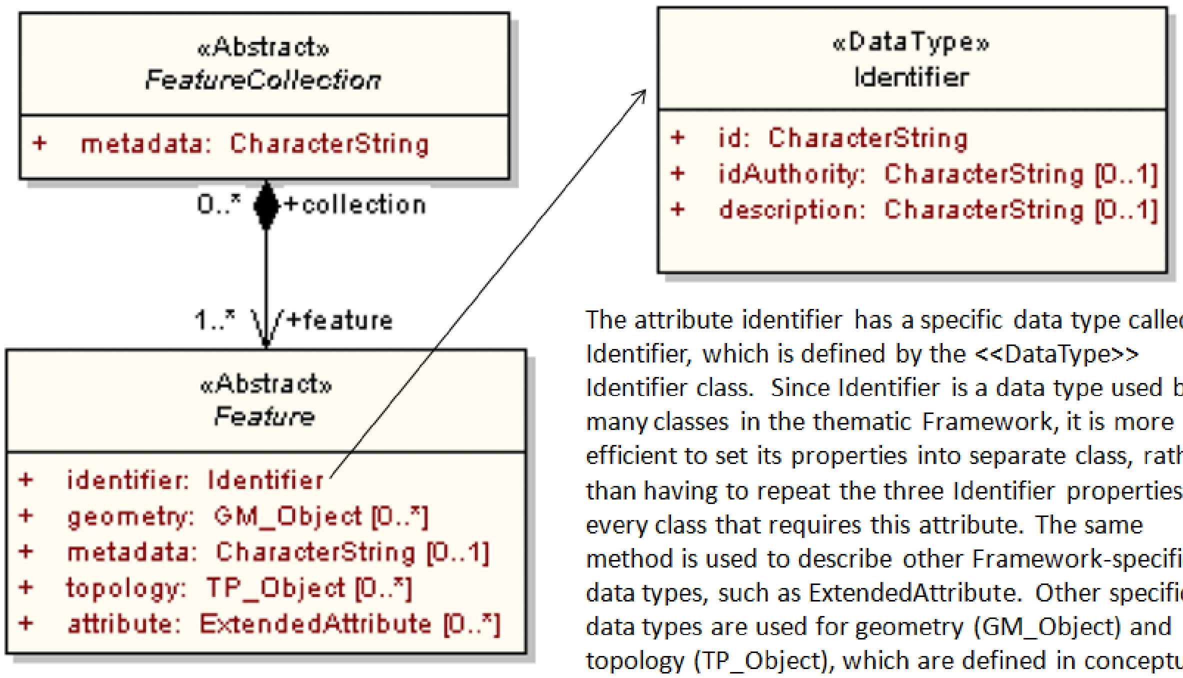
Another UML notation seen in Figure 2.2 is the line starting from <<Abstract>> FeatureCollection with a filled diamond pointing to <<Abstract>> Feature (see Figure 2.3). This filled diamond represents a relationship of composition. **Composition** is used when one class is a collection or container of other classes. A FeatureCollection is a collection of at least one or more features. How do you know that there must be at least one feature? There is a multiplicity notation next to the arrow, 1..* (see Figure 2.3). The multiplicity at the other end of the line, next to the filled diamond, is 0..*. This means you can have zero to many feature collections. Notice that there are also role names associated with the relationship between the two classes: role name +collection and role name: +feature (see Figure 2.3). The **role name** explains how the target object participates in a relation to the source object. Each role name is described in the data dictionary that accompanies each UML diagram.

In addition to the <<Abstract>> classes seen in Figure 2.2, the Framework Feature Model includes two other class stereotypes, <<Data Type>> and <<CodeList>>. Each of these stereotypes is discussed in the following subchapters.

<<Abstract>> stereotype
If the class name is italicized, it means it is an abstract class (also denoted by the stereotype <<Abstract>>). This means the class cannot be directly instantiated. This goes back to the example given earlier of a road or stream being an instance of a feature. Conceptually, a feature type can **instantiate** a road. However in the Framework application schema, you cannot create an instance of the <<Abstract>> class Feature because it is too generalized. However, specific *subtypse* of Feature can instantiate a road or stream. For instance, in the Transportation part of the standard, there is a subtype called TranFeature that is of the <<Feature>> class. A Tranfeature can instantiate a road, railroad, or other type of transportation.

All features in every part of the Standard are subclasses of the class Feature shown in Figure 2.2, and thus inherit its properties.

<<DataType>> stereotype
Notice that for the <<Abstract>> *Feature* class in Figure 2.3, some of the attributes have special data types. The attribute, *identifier*, has a data type Identifier. This special data type is fully described by the class stereotype <<Data Type>> Identifier which includes three specific attributes for Identifier.


«Abstract»
FeatureCollection
+ metadata: CharacterString
0..* +collection
1..* +feature
«Abstract»
Feature
+ identifier: Identifier
+ geometry: GM_Object [0..*]
+ metadata: CharacterString [0..1]
+ topology: TP_Object [0..*]
+ attribute: ExtendedAttribute [0..*]
«DataType»
Identifier
+ id: CharacterString
+ idAuthority: CharacterString [0..1]
+ description: CharacterString [0..1]

The attribute identifier has a specific data type called Identifier, which is defined by the <<DataType>> Identifier class. Since Identifier is a data type used by many classes in the thematic Framework, it is more efficient to set its properties into separate class, rather than having to repeat the three Identifier properties for every class that requires this attribute. The same method is used to describe other Framework-specific data types, such as ExtendedAttribute. Other specific data types are used for geometry (GM_Object) and topology (TP_Object), which are defined in conceptual models created for geographic information in general (ISO 19107), rather than just specifically for Framework.

**Figure 2.3**
**Using specific <<DataType>> classes to remove redundancy from the data model.**

Another specific instance of <<DataType>> is Extended Attribute. ExtendedAttribute allows additional attributes to be added to a dataset without compromising its Framework compliancy. This class provides for the description and transfer of additional ad hoc data content without requiring changes or extensions to the data schema. The repeatable structure may carry one or more additional attributes and their values for use in peer-to-peer transfer of unofficial feature properties.

**FAQ:** Why is the ExtendedAttribute important?

The ability to extend the Framework Standard beyond its defined content makes it adaptable, because technology is always changing and updating. Likewise, users have different needs and goals, so flexibility is important in the Standard as well.

<<CodeList>> stereotype
A <<codelist>> is a stereotype used to describe an open or flexible enumeration – in other words, a list of acceptable values that is not exhaustive. Additional values not listed may be acceptable. Codelists are useful for expressing a list of standard values to choose from. They provide the correct spelling of the values and usually also provide a definition or description of the value.

The Framework Feature Model includes two codelists, ResourceType and DataType, shown in Figure 2.4.

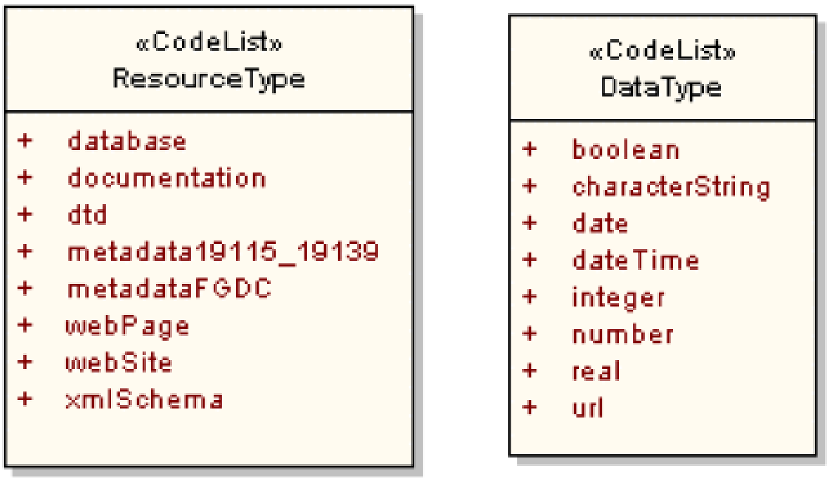


**Figure 2.4**
**Two codelists used in the Framework Feature Model (detail of Figure 3 in Part 0: Base document).**

Definitions for all the values in these two codelists are provided in Tables 3 and 4 in Part 0: Base document.

The <<codelist>> ResourceType is provided to classify the types of external resources (e.g. database, documentation, metadata, webpage) that can be used to describe the ExtendedAttribute.

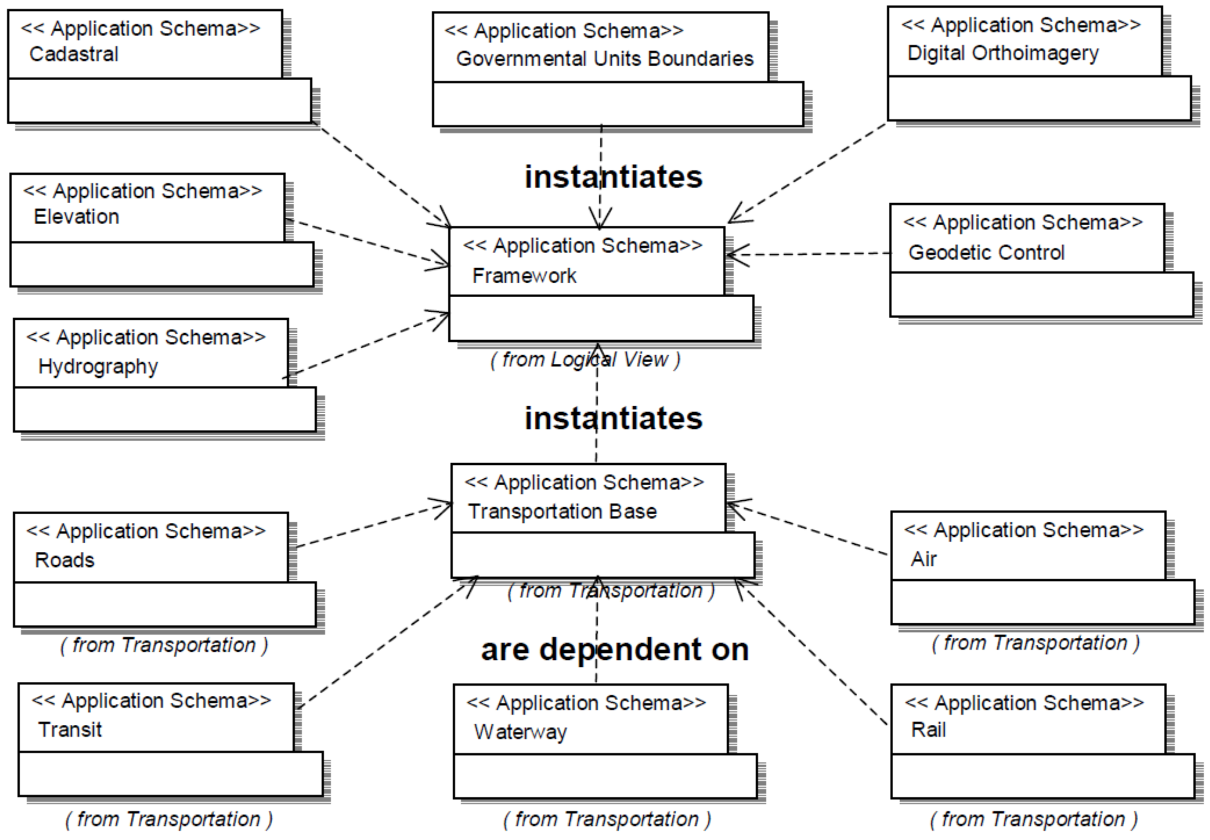
The <<codelist>> DataType is used to provide a list of acceptable data types (e.g. Boolean, CharacterString, Date). The name of this <<codelist>> is confusing, since "data type" has already been defined in general terms and as another specific stereotype, <<DataType>>. The difference to keep in mind is the purpose of the <<DataType>> stereotype versus the <<Codelist>> stereotype. The <<DataType>> stereotype defines a new data type specific for the purposes of Framework, e.g. Identifier. The <<Codelist>> DataType provides a list of general data types that are not specific just to Framework, but specifies the exact spelling that should be used to identify a particular type ("characterString" instead of "text" or "string").

**FAQ:** Why is the use of enumerations and codelists important to Framework?

One of the greatest barriers to integrating data from different sources is semantics (i.e., different terms are used by different organizations for the same data). The use of standardized terms, selected from enumerations and codelists provided in each part of the Framework Standard, will allow applications to recognize specific data for specific purposes and increase interoperability. For instance, the Governmental Units part of the Framework Standard includes a codelist called "GovernmentalUnitType" which includes names and definitions for the most common instances of governmental units. How should a state label areas that are part of native American reservations? "American Indian" "tribal" "native" are common terms; "lands" is a term frequently synonymous with reservation, as in "tribal lands". To avoid confusion of semantics, the Framework codelist provides two official terms to choose from: "americanIndianReservation" and "americanIndianTribalSubdivsion" with detailed descriptions of each (and additional options for special cases in Alaska).

#### 2.3.1.4 *Model integration*

Figure 2.5 is a UML model that shows how the application schemas for each thematic part are dependent on the overall application schema for Framework. More detailed UML models are provided in each thematic part (parts 1-7).



<< Application Schema>> Cadastral
<< Application Schema>> Governmental Units Boundaries
<< Application Schema>> Digital Orthoimagery
instantiates
<< Application Schema>> Elevation
<< Application Schema>> Framework
( from Logical View )
<< Application Schema>> Geodetic Control
<< Application Schema>> Hydrography
instantiates
<< Application Schema>> Transportation Base
( from Transportation )
<< Application Schema>> Roads
( from Transportation )
<< Application Schema>> Air
( from Transportation )
are dependent on
<< Application Schema>> Transit
( from Transportation )
<< Application Schema>> Waterway
( from Transportation )
<< Application Schema>> Rail
( from Transportation )

**Figure 2.5**
**Dependencies among models specified in the thematic parts (reproduced from Figure 2 in the Base Document)**

All of the thematic application schemas are dependent on the <<Application Schema>> Framework, which corresponds to Figure 2.2 in this document (or Figure 3 in the Base Document). The dependency means that all thematic data models inherit the properties identified in the <<Application Schema>> Framework.

In addition, UML models representing application schemas of the five sub-parts of the Transportation theme are dependent on the application schema for Transportation base.

#### 2.3.1.5 *Establishment of identifiers*

This requirement establishes the need for permanent identifiers (usually a unique number) for each data record associated with a feature type. This identifier can be used to distinguish between similar values in different datasets. For instance, two adjacent counties may have different roads with the same name, or two unrelated streams with the same name.  Permanent identifiers may also be accompanied by an optional identifier authority (commonly known as “namespace.”)

Each theme has a different prescription for how the permanent identifier and namespace are provided. For example, the Geodetic Control part of the Standard indicates that the permanent identifier can be the unique database identifier of the organization that creates/maintains the data. The namespace is the organization’s identifier (for example, abbreviation) for the organization who assigned/maintains the permanent identifier. The unique identifier allows traceability of each data point back to the organization and to other data held by that organization about the point.

On the other hand, the Governmental Units describes the permanent identifier merely as an identifier assigned to the unit, without indicating where the identifier originates. The identifier authority (e.g. namespace) is the "official" feature name, and is mandatory in this particular thematic part of the standard. Additionally, the namespace associated with the identifier must be the name of the geographic area feature from the Geographic Names Information System (GNIS) if that geographic area feature is present in the GNIS database.

The Base Document states that policies may be developed within a community for assigning namespaces and permanent identifiers to features and expressing equivalencies among features that have been assigned different namespaces and, therefore, different identifiers, which may be permanent. If there is no standard way to create and manage identifiers, users may develop their own schema and include its description in the dataset metadata.

Perhaps more so than any of the other parts of the Framework Standard, the Hydrography part (part 6), addresses specific needs for permanent identifiers:

> Permanent identifiers allow sharing of data in a distributed environment. Permanent identifiers are required to support maintaining entities over time (for example, exchange of updates); to support maintaining entities across multiple representations (for example, scale and/or dimension); to support maintaining associations to linked data; and to describe associations among local and external data entities, such as water quality or ecological surveys.

While the Hydrography part discusses specific implementations of the permanent identifier for each of these scenarios, many of the other thematic parts also imply similar implementations that require permanent identifiers and managing authorities. The following is also excerpted from the Hydrography part to describe the importance of managing authorities:

> Any permanent identifier scheme requires an authority to manage the included features. The model also allows assignment of local identifiers to server as temporary identifiers or as crosswalks to permanent identifiers. If a permanent identifier does not exist, the model requires at least a unique temporary identifier be assigned for the purpose of the exchange. All feature identifiers must be unique in the context of the transfer and within an authority’s coding system.
>
> The responsibilities of an authority include: recognition as a maintenance authority within the community, ability to assign, update, manage and publish identifiers, and to assure that identifiers are discoverable by users within a reasonable timeframe of their registration.

Users may also consider publishing their namespaces and permanent identifiers with metadata registries. As an example, the North American Profile of ISO 19115: Geographic Information – Metadata has a metadata register for identifying specific metadata classes and associated codelists.

Digital orthoimagery and elevation are not composed of features in the same sense as the other Framework themes, but their specific application schemas inherit the Framework class <<feature>> at a high level. For instance, a range of elevation values for a specific geographic extent is considered a Framework feature, and as such requires an identifier.

### 2.3.2 Data Dictionary

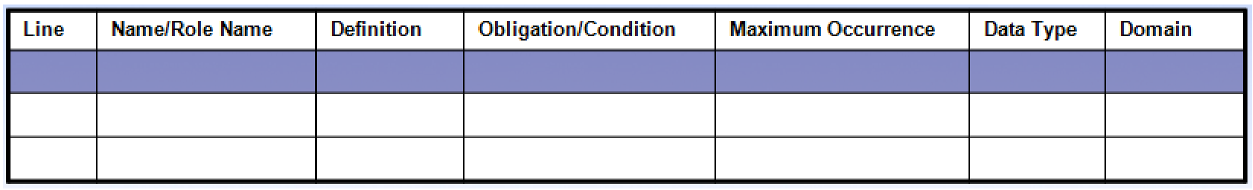
For data to be compliant with the Framework Standard, it needs to conform to the data dictionary table(s) provided for the thematic part of the standard which applies to the data in question. Each table includes one or more entities, and each entity may contain one or more elements. Not all of the entities and their elements are mandatory (required for conformance). Some are conditional or optional.

An application schema is described both by UML model in the form of a class diagram and a data dictionary. For every UML model found within the various parts of the Standard, there is a corresponding data dictionary table. The data dictionary table describes the characteristics of the UML model diagrams in more detail.

Subchapter 2.3.1.4, Model integration, shows how the application schemas for each of the seven themes are dependent on the Framework application schema. Each of these thematic application schemas are provided in more detail in the thematic parts of the Standard (parts 1-7). Therefore, Framework compliant data for Hydrography, for instance, must conform to the mandatory entities and elements of the data dictionaries of the Hydrography application schema, which means that it also conforms to the data dictionary for the Framework application schema in the Base Document. The entities and elements in the data dictionaries of the Hydrography schema are more detailed instances of the abstract entities and elements identified in the Framework schema.

The data dictionary table is composed of seven columns and a number of rows that corresponds to the number of classes, attributes of classes, and roles associated with classes in the UML model (see Figure 2.6). There are two types of rows in the table: entity (a shaded row) and element (an unshaded row).

- Each class in the UML model corresponds to a data dictionary entity (a shaded row).
- Each class attribute in the UML model corresponds to a data dictionary element (the unshaded rows beneath the entity row).
- Each role name in a UML model also corresponds to a data dictionary element (unshaded row).
- The entities and elements within the data dictionary are defined by six attributes based on those specified in ISO/IEC 11179-3 for the description of data element concepts



| Line | Name/Role Name | Definition | Obligation/Condition | Maximum Occurrence | Data Type | Domain |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |

**Figure 2.6**
**Basic format of a data dictionary table.**

Name/Role name
The name/role name is a label assigned to a data dictionary entity or to a data dictionary element. Entity names are unique within a data theme, and start with a capital letter. Element names start with a lower case letter. Combinations of the entity and element names (example: Dataset.name) are therefore unique within a data theme. Role names are used to identify the roles of the classes at the ends of a model association and are preceded by the term "Role name" followed by a colon to distinguish them from other types of data dictionary elements.

Definition
The definition is the entity or element description.

Obligation/Condition
This is specified only for rows that contain elements, and consists of three values: M (mandatory), C (conditional) and O (optional).

- Mandatory (M) – indicates that the element must be populated with a value
- Conditional (C) – the element must be populated based on certain conditions. More information about particular instances which warrant Conditional are described in section 7.4.4.3 of the Base Document.
- Optional (O) – the element may be populated, an option for those looking to fully document their data. Use of this common set of defined elements will help promote interoperability among framework data users and producers.

Optional entities may have mandatory elements. If the optional entity is used, the mandatory elements shall be used. An example of an optional entity can be seen in the UML model for Governmental Units. Governmental units can be described spatially in two different ways. One way (mandatory) is an instance of <<Feature>> called GeographicArea. Optionally, a <<Feature>> called Boundary may also be used. If Boundary is used, then its mandatory element Identifier must be included.

Entities (e.g. classes) may also be mandatory or optional, though this is not expressed in the data dictionary. In the UML diagram, a class is required if it has no multiplicity associated with it (default) or if its multiplicity is [1..*]. A multiplicity of [0..n] indicates that it is optional, since 0 instances are possible.

Maximum Occurrence
Used only in rows that contain elements, maximum occurrence specifies the maximum number of instances the element may have. Single occurrences are shown by .1.; unconstrained number of instances are represented by an asterisk .*.. Fixed number occurrences, other than one, are allowed and will be represented by the corresponding number (that is, .2., .3. .and so on). If the

element is a role name, then the maximum occurrence shall apply to the element indicated by the Data Type.

Data type
Specifies a set of distinct values for representing the elements. Examples include integer, real, CharacterString, DateTime, Boolean and Framework-specific data types such as <<Feature>>, <<DataType>>: Identifier or <<DataType>>: ExtendedAttribute
For elements that are role names (defining a role played by a class in an association with another class), the data type attribute will list the name of the class. These classes are generic types (as indicated by the <<abstract>> stereotype) that do not infer an implementation.

Domain
For an entity, the domain indicates line numbers of the elements of that entity in the table. For an element, the domain specifies the values allowed. "Unrestricted" indicates that no restrictions are placed on the data type of the element. Sometimes the domain will refer to the name of a codelist, such as ResourceType, or the name of an enumeration. Code lists provide a list of potential values, although additional values can be used. Enumerations provide a non-extensible list of values.

### 2.3.3 Metadata

For data to be compliant with the Framework Standard, it must have metadata that conforms to at least the minimal set of mandatory elements of either ISO 19115, Geographic Information - Metadata, or FGDC-STD-001-1998, Content Standard for Digital Geospatial Metadata version 2.0. Efforts are underway to finalize a new national (ANSI) standard - *The North American Profile (NAP) of the ISO 19115: Geographic Information – Metadata*. The NAP is also an acceptable form of metadata for Framework data.

While the Framework Standard requires a minimal set of mandatory metadata elements, it also states that more extensive metadata should be provided. The FGDC provides many resources for assistance with metadata implementation. The SDI Cookbook (Nebert 2004) also provides an excellent resource for understanding the benefits of metadata and its crucial relationship to geospatial data, as well as a discussion of implementation approaches to metadata.

**FAQ:** Specifically what is the "more extensive metadata" that should be provided?

Mandatory elements are primarily focused on describing the data for purposes of data discovery. More extensive metadata includes the sources of the data, processes used to create or update the data, and quality of the data. Providing more extensive metadata allows users to match data to their needs. Well-crafted metadata facilitates the search and collection process while alleviating some of the burden on the user to assess quality and applicability of data. The more metadata content there is for a product, the more it can support the user's determination of its reliability, quality, and accuracy. Metadata is intended to be of value to the producer as well as to the user; with quality metadata producers are able to maintain data integrity through different versions and also through workforce changes.

The following text is taken from section 7.5.2, Associating metadata entry with data transfer, from the Base Document:

> The mechanism used to associate a structured metadata entry with a data transfer is not explicitly declared in the Framework Data Content Standard due to possible complex dependencies on either the structure of FGDC or ISO metadata being used. It is the intention of the standard to logically insert the appropriately structured metadata from either standard wherever the class attribute “metadata” occurs. The implementation of this capability may be specified in the implementation annexes as referenced to external metadata schemas in the appropriate implementation or programming environment.

**FAQ:** What is meant by implementation annexes with regard to external metadata schemas?

A metadata schema is metadata structured in a standardized format. A single metadata schema, such as ISO 19115, may be expressed in different markup or programming languages, each of which requires a different syntax. A metadata schema will often include an implementation annex, which describes how the schema may be implemented in a markup or programming language. The ISO 19115 metadata schema does not have a requirement that metadata be implemented in XML, but that is one possible type of implementation. While the syntax is not strictly part of the metadata schema, the data will be unusable unless the encoding scheme understands the semantics of the metadata schema. The encoding allows the metadata to be processed by a computer program.

# 3. Thematic parts of the Framework Data Content Standard (parts 1-7)

The following subchapters 3.1 through 3.7 cover the seven thematic parts of the Framework Standard, in the following order: Cadastral, Digital Orthoimagery, Elevation, Geodetic Control, Governmental Units, Hydrography and Transportation.

The seven parts are organized in a similar manner, though some of the sections have different headings. Much of the wording used in the Introduction, Scope, Purpose, Conformance, and Maintenance Authority sections of the thematic parts is similar. Subchapter 1.5 of this Guidance Document discusses the information common to all or most of the parts of the Framework Standard. The following subchapters include only information specific and unique to each thematic part.

This document is intended to be used as a reference alongside the thematic parts of the Framework Standard. In some cases, portions of UML class diagrams and other figures from the thematic parts are reproduced in this document with additional notations for clarification. Chapter 3 summarizes information from the thematic parts, though there are instances which include more in-depth descriptions and/or detailed clarifications (e.g., examples for classes that are particularly complex).

The content is organized to complement the material found in the Standard, by presenting it in a more accessible manner. Using information from the thematic parts often requires moving back and forth between different sections of the Standard. For instance, from the section that contains the data model and its data dictionary, it may be necessary to flip back to a previous section that contains definitions of terms, and then flip to another section that contains codelists with values, and then to yet another section that contains implementation examples. This guidance document attempts to organize all the relevant information (definitions, structures, values from codelists and examples) in a more cohesive unit.

## 3.1 Framework Data Content Standard Part 1: Cadastral

### 3.1.1 Introduction

Cadastral data are defined in this part of the Standard as "the geographic extent of the past, current and future rights and interests in real property including the spatial information necessary to describe that geographic extent." Cadastral data can include surveys and legal description frameworks such as the Public Land Survey System as well as parcel-by-parcel surveys and descriptions.

Of the seven NSDI Framework themes, the Cadastral part is unique in the respect that it is a profile of another standard, the Cadastral Data Content Standard, developed by the Federal Geographic Data Committee (FGDC-STD-003). The Cadastral Data Content Standard contains the data intended to support the automation and integration of publicly available land records information. Part 1 of the Framework Data Content Standard is designed to work with the Cadastral Data Content Standard toward establishing a common baseline for the semantic content of cadastral databases for public agencies and private enterprise, toward more efficient data exchange and integration.

Part 1 includes only the minimum data necessary to facilitate locating the existence of parcel-level information and identifying the source. This minimal set of data, along with the appropriate metadata, will provide the information describing how and where to get the data needed to support applications. Example applications of cadastral data for homeland security, citizen query and access, and real estate records require the Cadastral Data Content Standard in its entirety. For access to the FGDC Cadastral Data Content Standard, please see the *FGDC Standards publications webpage*.

For more details, refer to section 1 of Part 1: Cadastral. Annex B of Part 1 contains a sample diagram of cadastral data. The *FGDC online training materials webpage* has materials for download (PowerPoint slides) specific to Part 1: Cadastral. These materials include descriptions and graphic examples of different types of cadastral data, as well as links to several websites that describe or serve cadastral data.

This part of the Guidance Document summarizes information about cadastral data content specific to the Part 1: Cadastral document, and does not attempt to summarize information about the more detailed data models in the Cadastral Data Content Standard.

### 3.1.2 Requirements

The Requirements (section 5) of the Framework Standard, Part 1: Cadastral includes a UML class diagram of the Cadastral application schema required for locating the existence of cadastral data, along with a data dictionary tables that describes the UML objects in more detail, and a codelist that includes standard values for owner types.

Figure 3.1.1, the Parcel class, shows the two main <<feature>> classes from the complete UML class diagram (Figure 1, Cadastral UML model) from Part 1 of the Framework Standard. The selected classes are shown for simplification, omitting other <<DataTye>> and <<CodeList>> classes.

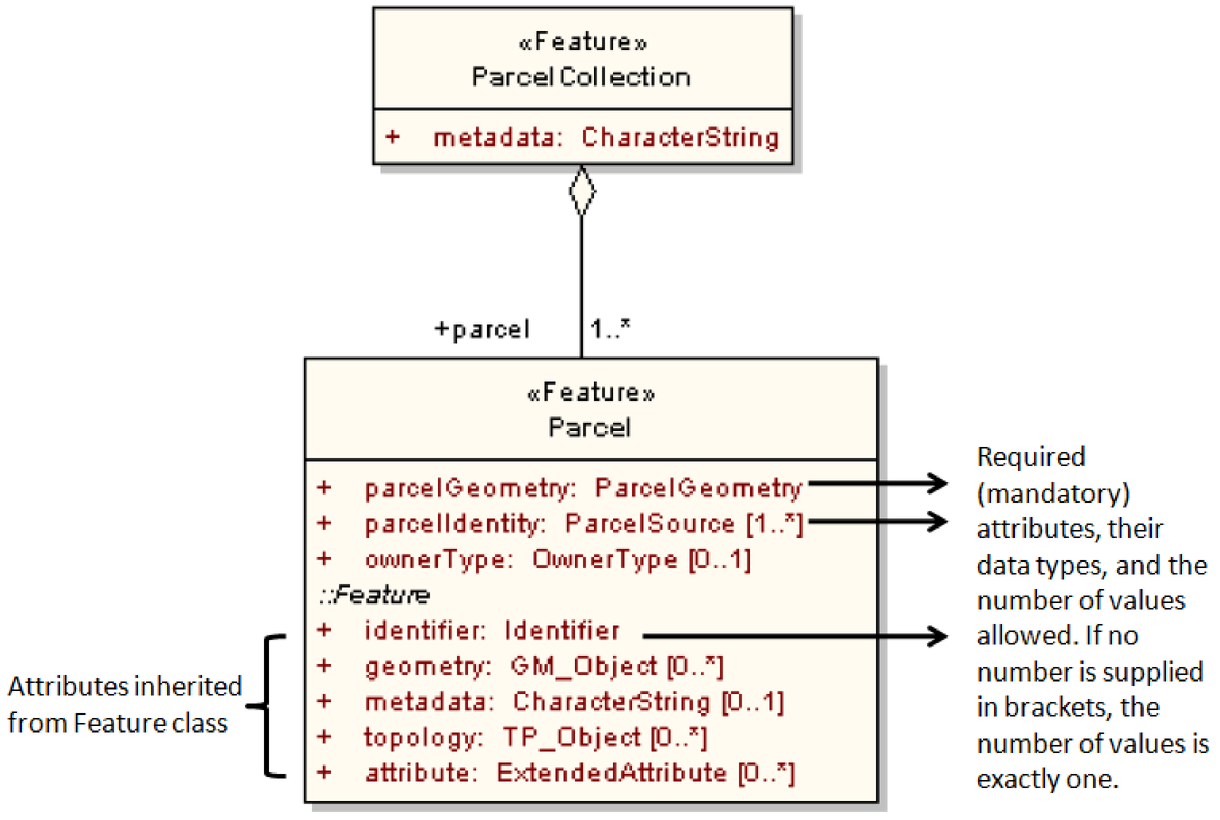

«Feature»
Parcel Collection
+ metadata: CharacterString
+parcel
1..*
«Feature»
Parcel
+ parcelGeometry: ParcelGeometry
+ parcelIdentity: ParcelSource [1..*]
+ ownerType: OwnerType [0..1]
::Feature
+ identifier: Identifier
+ geometry: GM_Object [0..*]
+ metadata: CharacterString [0..1]
+ topology: TP_Object [0..*]
+ attribute: ExtendedAttribute [0..*]
Required (mandatory) attributes, their data types, and the number of values allowed. If no number is supplied in brackets, the number of values is exactly one.
Attributes inherited from Feature class

**Figure 3.1.1**
**The Parcel feature class, the main class in the Cadastral UML model**
**(detail from Figure 1, Cadastral UML Model, in Part 1: Cadastral).**

The <<Feature>> class ParcelCollection at the top of Figure 3.1.1 represents a collection of parcel features requested for data transfer along with metadata describing the date and content of the transfer. See subchapter 2.3.1.3 for more information about the FeatureCollection class that is the parent class of ParcelCollection.

The <<Feature>> class Parcel is the main class to convey cadastral information for data discovery. It is a subtype of the <<Abstract>> Feature class from the overall Framework application schema specified in the Base document, which means that Parcel features inherit one required attribute, *identifier*, and several optional attributes.

In addition to its inherited attributes, a parcel feature has three attributes, two of which are required. The attribute *parcelGeometry* is required and can either be a polygon representation of a parcel location, or a centroid (point that is located at the geometric center of the parcel). The other required attribute is *parcelIdentity*, and there can be more than one instance of this attribute.

Some confusion may exist at this point as to the difference between the *identifier* and *parcelIdentity* attributes, especially since there can be more than one *parcelIdentity,* which seems to violate the principle of a unique identifier.

- *identifier* is a unique identifier of a feature used for data discovery purposes. This may be thought of as an artificial feature or even just a placeholder for the actual parcel data. For discovery purposes, it is not necessary to include the actual parcel data (especially if it is complex in nature). In fact, the Parcel feature can actually encompass more than one representation of a real parcel. For instance multiple representations of a parcel can be a result of different sources, or from one source but including past versions of the parcel, a current (primary) version and even future versions.
- *parcelIdentifier* is a way of identifying a real parcel feature used for cadastral applications, and is always accompanied by three pieces of information:
    - *parcelID*: the unique identifier for the parcel used by the source that created or maintains the real parcel data
    - *sourceId*: the linkage to the source agency or organization
    - *primary*: a Boolean attribute which if true means that the current record describes the primary parcel.

In addition to the required attributes, an optional *ownerType* attribute is also associated with a Parcel feature for data discovery purposes. A CodeList called OwnerType is provided in order to standardize common ownership terms, such as federalGovernment, localGovernment, private, tribalNation and so forth. A codelist is not an exhaustive list and allows for additional values to be used, but implementers should use values from the list if appropriate for their data instead of substituting their own variation of the same value.

Table 3.1.1 shows all the required attributes and example values for a Parcel feature to be used for data discovery purposes.

**Table 3.1.1.**
**Example of a Parcel feature that encompasses two representations of a real parcel.**

| Parcel attributes | Specific DataType attributes or Codelist | Example value(s) | |
|---|---|---|---|
| **Identifier** | **id** (Framework:Identifier DataType) | 12 | |
| **parcelGeometry** | ParcelGeometry DataType | GM_Point[1] | |
| ownerType | OwnerType codelist | localGovernment | |
| **parcelIdentity** (uses ParcelSource datatype) | **parcelId: id** (Framework:Identifier DataType) | 39-063a | 39-063b |
| | **sourceId: id** (Framework:Identifier DataType) | SANDAG | SANDAG |
| | **Primary** | FALSE | TRUE |

[1] GM_Point (and the other option, GM_Polygon) are described in ISO 19107: Geographic Information - Spatial Schema

Note that in Table 3.1.1, several of the Parcel attributes (*identifier*, *parcelGeometry*, and *parcelIdentity*) use DataType classes which have their own distinct set of attributes. This reflects a method of organization to reduce the number of times attributes need to be listed in classes and data dictionaries, but in the actual data all the required attributes must be included.  This means that a Parcel feature shows only three required (mandatory) attributes in its data dictionary (see Table 1, lines 5,6 and 8 in Part 1: Cadastral), but when all the DataType attributes are included with example values in Table 3.1.1, there are a total of five required attributes.

The Cadastral data model (for data discovery purpose) may be implemented various ways. As an example, if the data model is implemented in a relational database structure, it might require two separate tables in order to handle the multiple values associated with some attributes. The primary table would include the Parcel attributes identifier, *parcelGeometry*, *parcelIdentity* and the optional *ownertype*. The second table would include the attributes *parcelIdentity*, *parcelId*, *sourceId*, and *primary*. The attribute in common between both tables is *parcelIdentity*. A query on *parcelIdentity* in the second table would allow a user to find all the instances of multiple representations of a real parcel, along with their sources so the desired data can be retrieved. Ideally, the optional *metadata* attribute would also be included, so the user would have more information about the date and accuracies associated with the multiple representations.

## 3.2 Framework Data Content Standard Part 2: Digital Orthoimagery

### 3.2.1 Introduction

Digital Orthoimagery combines the image characteristics of an aerial photograph, digital image, or other remotely-sensed data with the geometric qualities of a map. Digital orthoimages encode the optical intensity of sensed radiation in one or more bands of the electromagnetic spectrum as discrete values in an array of pixels that model the scene observed. Unlike a typical aerial photograph, distortions due to relief displacement (hills, stream valleys, or buildings), camera lens, and aircraft attitude have been removed so that all ground features are shown in their correct ground positions. This permits direct measurement of distances, areas, angles, and the detailed portions of ground features that are typically omitted or generalized on traditional maps. Photo enlargements, simply rectified and rubber sheeted images are not orthoimages and do not comply with the basic procedures involved in photogrammetry that produce accurate orthoimages.

In a digital format, orthoimagery fulfills a fundamental role as a geometrically accurate base map. Many geographic features, including some in other framework data themes, can be interpreted and compiled from an orthoimage.

Digital orthoimagery can come in a wide array of different formats and can be collected by a variety of sensors. The primary focus of this part is on images sensed in the visible to near infrared portion of the electromagnetic spectrum. However, images captured from other portions of the electromagnetic spectrum are not precluded.

This part of the Framework Data Content Standard specifies data content and logical structure for the description and interchange of framework digital orthoimagery. To a certain extent, it also provides guidelines for the acquisition and processing of imagery (leading toward the generation of digital orthoimagery), and specifies the documentation of those acquisition and processing steps.

Because of rapidly changing technologies in the geospatial sciences, this part covers a range of specification issues, many in general terms. Complete and accurate reporting of information relating to quality control is critical, as well as standards employed in testing orthoimagery data. This reporting is stored as metadata in either the FGDC-STD-001-1998 standard or the ISO 19115 standard. Part 2: Digital Orthoimagery includes a section specific to metadata, in order to stress its importance, with similar wording to subchapter 2.3.3 in this document.

Data transfer formats for digital orthoimagery are not specified in this part. Data producers are encouraged to employ ISO and ANSI standards for information exchange, but like all other parts of the Framework Standard, the emphasis is on data content and not on specific formats which could potentially limiting to interoperability. In order to support interoperability, producers of orthoimagery data shall provide detailed descriptions of the format used.

Like most of the other thematic parts, the Digital Orthoimagery part has a Requirements section (section 8) that describes the structure of the data and other requirements such as resolution, accuracy and production components. Two additional sections are used to address image rectification and restoration (section 9) and image mosaicking (section 10). The UML class diagrams for the Orthoimagery data model and its accompanying data dictionary is found in Annex A, followed by codelists and enumerations that provide standard attribute values.

A data example (in a similar format to Table 3.1.1 in this document) is provided in Annex B.

Additional information about control is provided in Annex C. Control data are surveyed ground points, which provided exact coordinates and elevations of known locations on the Earth's surface. They are used in the orthorectification process; this annex discusses three different methods used to acquire the necessary control for digital orthoimagery.

A bibliography is provided in Annex D, with documents that are relevant to digital orthoimagery as it is put forth in this part of the standard. In addition to these sources, the *FGDC online training materials webpage* has materials for download (PowerPoint slides) specific to Part 2: Digital Orthoimagery. These materials include other graphic examples of different types of orthoimagery data and collection methods, as well as descriptions and links to several websites that describe or serve orthoimagery data.

### 3.2.1 Requirements

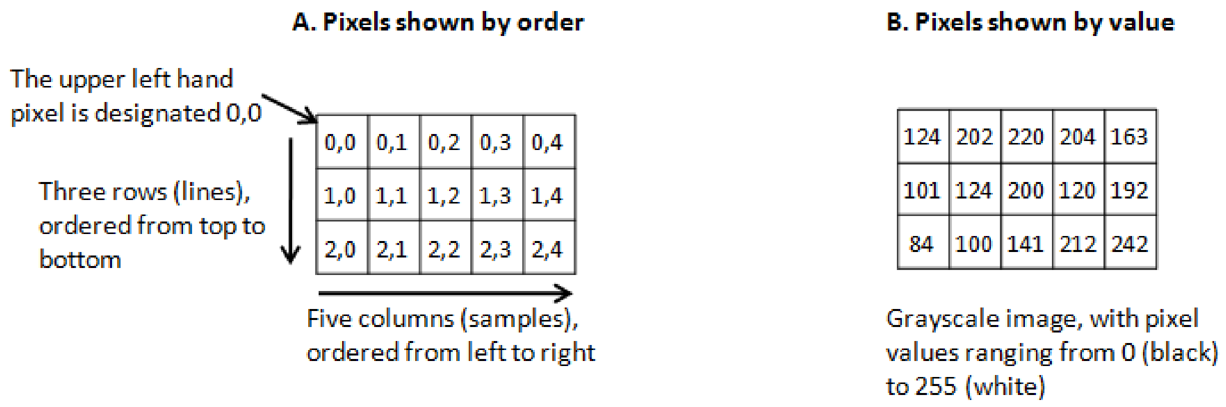
The following subchapters discuss five different requirements for digital orthoimagery: how it is structured, acceptable resolutions, coordinate systems and datums, accuracy, and production components.

#### *3.2.1.1 Structure*

The structure of Framework digital orthoimagery shall consist of images, each of which consists of a two-dimensional, rectangular array of pixels, as shown in Figure 3.2.1.

The ground area covered by each pixel, called ground resolution cells, determines the resolution of each pixel.

Part A of Figure 3.2.1 shows the order that pixels must be arranged in horizontal rows (lines) and vertical columns (samples). The order of the rows shall be from top to bottom; the order of columns shall be from left to right. The uppermost left-hand pixel shall be designated pixel (0,0).


A. Pixels shown by order
The upper left hand pixel is designated 0,0
Three rows (lines), ordered from top to bottom
Five columns (samples), ordered from left to right
B. Pixels shown by value
Grayscale image, with pixel values ranging from 0 (black) to 255 (white)

| | | | | |
|---|---|---|---|---|
| 0,0 | 0,1 | 0,2 | 0,3 | 0,4 |
| 1,0 | 1,1 | 1,2 | 1,3 | 1,4 |
| 2,0 | 2,1 | 2,2 | 2,3 | 2,4 |

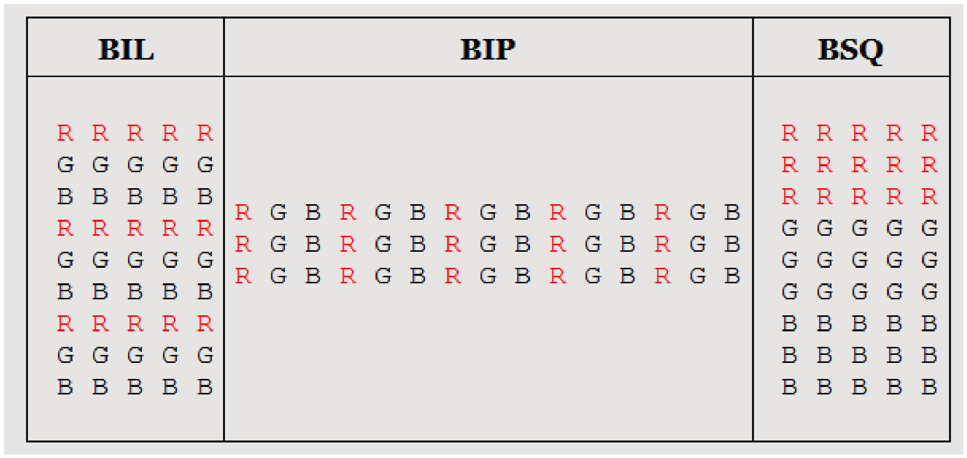
| | | | | |
|---|---|---|---|---|
| 124 | 202 | 220 | 204 | 163 |
| 101 | 124 | 200 | 120 | 192 |
| 84 | 100 | 141 | 212 | 242 |

**Figure 3.2.1**
**Two examples of an image 3 pixels high and 5 pixels wide, one showing how pixels are ordered and the other showing example values.**

Part B of Figure 3.2.1 shows the same 3 line, 5 sample image but in this example the pixels contain example values. Grayscale images record a value ranging from 0-255 in each pixel.

Images describing more than 1 band of electromagnetic radiation (natural color, color infrared, multi-band) shall be structured in one of three orders: band interleaved by line (BIL), band interleaved by pixel (BIP), or band sequential (BSQ). These are three different ways to express the grid structure in a simple binary file where lines in the file contain values corresponding to a row in the image.

Figure 3.2.2 represents the same-sized image as in Part B of Figure 3.2.1, three pixels high and five pixels wide, but in this case it is an example of an image with three bands: a red band, green band and blue band, which means each pixel will have three values represented by RGB. The other difference in Figure 3.2.2 is that instead of showing each pixel, it is showing how values are ordered in each line of a binary file for the three different orders: BIL, BIP or BSQ.



| BIL | BIP | BSQ |
|---|---|---|
| R R R R R | | R R R R R |
| G G G G G | | R R R R R |
| B B B B B | | R R R R R |
| R R R R R | R G B R G B R G B R G B R G B | G G G G G |
| G G G G G | R G B R G B R G B R G B R G B | G G G G G |
| B B B B B | R G B R G B R G B R G B R G B | G G G G G |
| R R R R R | | B B B B B |
| G G G G G | | B B B B B |
| B B B B B | | B B B B B |

**Figure 3.2.2**
**Three different ways to structure an image with three bands. Each pixel has three values corresponding to red, green, blue (RGB). Actual RGB values are not shown.**

The image shall have equal line (row) and column lengths, resulting in a rectangular image. This may be accomplished by padding with over-edge image or non-image pixels, that have a digital number (DN) equal to zero (black or no reflectance), in order to make the number of rows equal to the number of columns. The bounding coordinates of the image shall be documented in the metadata. For images that contain over-edge coverage or are padded with non-image pixels, descriptions of both the specific area of interest and any over-edge coverage shall be documented in the metadata.

This part of the Framework Standard places no constraints on the geographic extent of an orthoimage.

#### *3.2.1.2 Resolution*

When referring to orthoimagery, three different definitions of resolution are important: spatial, spectral, and radiometric.

Spatial resolution is the smallest unit which is detected by a sensor. Often expressed as pixel resolution or ground sample distance (GSD), it defines the area of the ground represented in each pixel in X and Y components (see Figure 3.2.3). For the purpose of this part, Framework digital orthoimages shall have a GSD of 2 meters or finer. Section 8.2.1 of Part 2: Digital Orthoimagery contains more information about spatial resolution and rules pertaining to resampling images to different spatial resolutions.

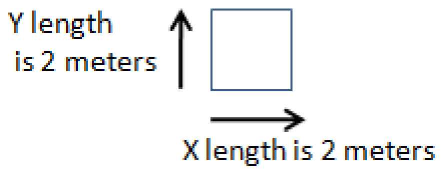


**Figure 3.2.3**
**A single pixel from an image with a ground cell resolution (GSD) of 2 meters, showing that the GSD is based on the x and y dimension of the pixels. The arrows also represent the x and y offset vectors, which are required attributes (see subchapter 3.2.4 for more about required attributes).**

Spectral resolution describes a sensor's sensitivity to a particular wavelength band or bands. For the purpose of this part, the focus for framework orthoimage will be on images sensed in the visible to near infrared portion of the electromagnetic spectrum, 0.4 to1.0 micrometers. However, this does not preclude images captured from other bands.

Radiometric resolution is the sensitivity of a detector to measure radiant flux that is reflected or emitted from a ground object. Relative radiance from the ground resolution cells shall be described by numerical representations (digital numbers (DNs) or brightness values) of reflected radiance amplitudes. The cell value for a single band shall be recorded as a series of binary digits or bits, with the number of bits per cell determining the radiometric resolution of the image. For instance, a radiometric resolution which records values from 0-255 requires 8 bits; a higher radiometric resolution which records values from 0 to 4095, requires 12 bits to store these larger numbers. Section 8.2.3 of Part 2: Digital Orthoimagery contains more information about radiometric resolution.

#### *3.2.1.3 Coordinate Systems and reference datums*

A common method for referencing coordinate positions on the Earth is essential for integrating geospatial data. Orthoimagery can be represented in a geographic coordinate system or a projected coordinate system , such as Universal Transverse Mercator (UTM) or State Plane Coordinate Systems (SPCS).  Every coordinate system must include a datum, a set of reference points on the Earth's surface against which position measurements are made. The North American Datum of 1983 (NAD83) or World Geodetic System 1984 (WGS84) datum shall be used as the horizontal datum for framework digital orthoimagery.

The regular spacing of a grid requires that only one point be referenced to a coordinate, usually the grid's origin. For digital orthoimagery, the origin pixel (the pixel denoted in position 0,0 in Part A of Figure 3.2.1) must have its geographic position described by a set of x,y coordinates which correspond to whichever coordinate system (geographic or projected) the image is in. The position of the all the pixels in the image can be inferred from the origin's coordinates, because of the regular spacing of the grid. However, additional coordinates may be provided for georegistration.

#### *3.2.1.4 Accuracy requirements*

Accuracy of new or revised spatial data shall be reported according to the National Standard for Spatial Data Accuracy (NSSDA) [FGDC-STD-007.3-1998]. The NSSDA implements a statistical and testing methodology for estimating the positional accuracy of points in digital geospatial data, with respect to georeferenced ground positions of higher accuracy. This reporting methodology provides a common language for reporting positional accuracy so that users can evaluate datasets for fitness of use for their applications. Section 8.5 of Part 2: Digital Orthoimagery contains more information about how accuracy is reported in the metadata for digital orthoimages.

#### *3.2.1.5 Production components*

Since digital orthoimages form the basis for creating many other types of data, it is important that they are created through a true displacement rectification process, also called differential rectification or orthorectification. The orthorectification is a point-by-point correction of the scale and relief displacements normally resulting from variations in elevation between the aircraft and the topography over which the aircraft or satellite flies.

Production components used in the orthorectification process include the image sources, elevation data, control (positions on the ground that have been surveyed for very precise locations), and camera or sensor calibration data. Section 8.6 of Part 2: Digital Orthoimagery contains more information about the requirements for each of these components.

### 3.2.2 Additional considerations for digital orthoimages

Section 9 of Part 2: Digital Orthoimagery contains information about image rectification and restoration. processes for correcting distortions and degradations that result from image acquisition or production all accept raw (or unprocessed) imagery that contain some degree of error in geometry (geometric distortion) and in the measured brightness values of the pixels (radiometric distortion). Please refer to this section in Part 2: Digital Orthoimagery for more details.

Section 10 of Part 2: Digital Orthoimagery contains a statement about image mosaicking. Single orthoimages are commonly created through the mosaicking of multiple images and many producers go through extensive image processing steps to attain a “seamless” appearance. This part of the Framework Standard does not discuss mosaic procedures nor will it prescribe the degree of quality for the appearance of mosaicked orthoimages. However, all the images that comprise the source of a mosaicked image shall be documented in the metadata field.

Section 11 of Part 2: Digital Orthoimagery contains a statement about data transfer formats. As discussed earlier in subchapter 3.3.1, data transfer formats for digital orthoimagery are not specified in this part. Endorsing one particular format would negate the goal of interoperability that the Framework Data Content Standard seeks to achieve.

Section 12 of Part 2: Digital Orthoimagery contains a statement about metadata, in particular that good metadata is need to support the exchange and use of geospatial data. Please see subchapter 2.3.3 of this document for more information about metadata and its importance.

### 3.2.3 Orthoimagery application schema (data model)

Figure 3.2.5 is a detail of Figure A.1 from Part 2: Digital Orthoimagery, showing only the two major classes in the Orthoimagery class diagram.

Classes prefixed with a CV_ are classes from ISO 19123: Geographic information – Schema for coverage geometry and functions. All of the attributes are mandatory, and in implementation this information is often included in header files or metadata files accompanying orthoimages.

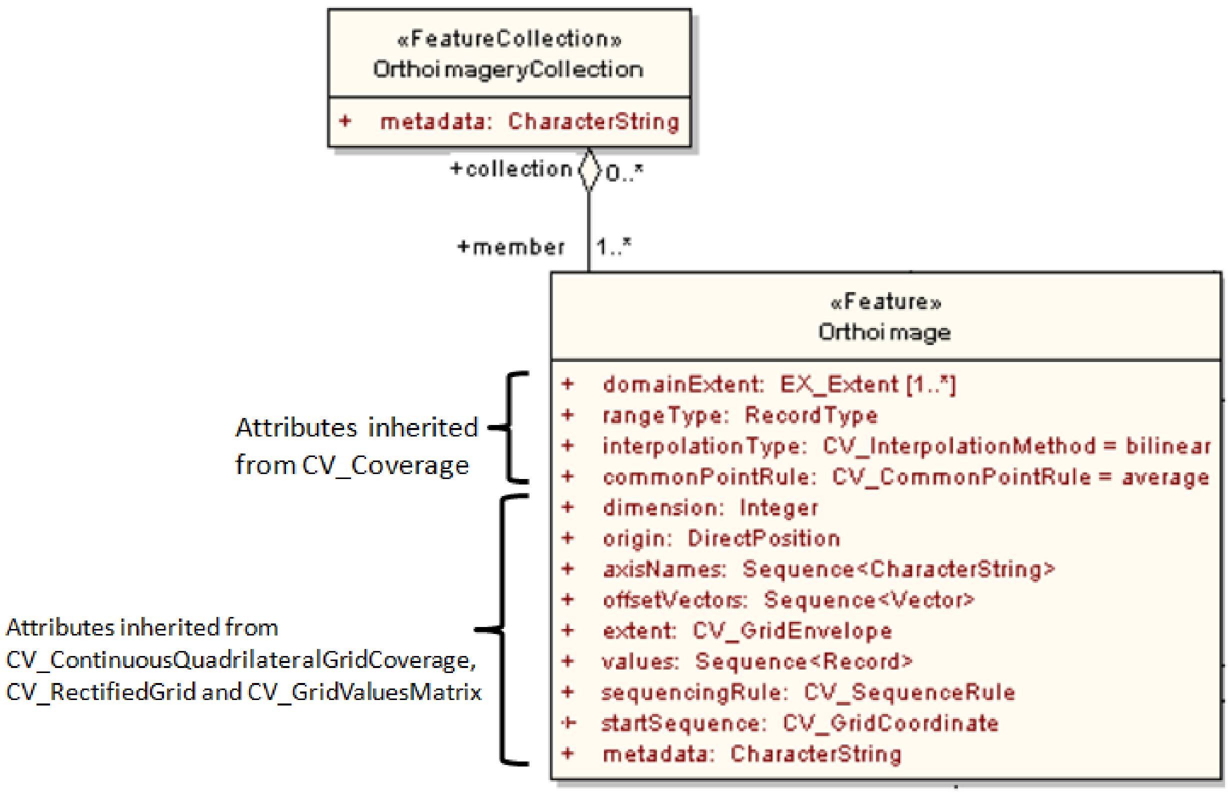

«FeatureCollection»
OrthoimageryCollection
+ metadata: CharacterString
+collection
0..*
+member
1..*
«Feature»
Orthoimage
+ domainExtent: EX_Extent [1..*]
+ rangeType: RecordType
+ interpolationType: CV_InterpolationMethod = bilinear
+ commonPointRule: CV_CommonPointRule = average
+ dimension: Integer
+ origin: DirectPosition
+ axisNames: Sequence<CharacterString>
+ offsetVectors: Sequence<Vector>
+ extent: CV_GridEnvelope
+ values: Sequence<Record>
+ sequencingRule: CV_SequenceRule
+ startSequence: CV_GridCoordinate
+ metadata: CharacterString
Attributes inherited from CV_Coverage
Attributes inherited from CV_ContinuousQuadrilateralGridCoverage, CV_RectifiedGrid and CV_GridValuesMatrix

**Figure 3.2.5**
**The OrthoImage class (detail of Figure A.1 in Part 2: Digital Orthoimagery)**

A data example including all required attributes is provided in Annex B of Part 2: Digital Orthoimagery. Each of the attributes are described below, along with example values taken from Annex B. The example image is in a geographic coordinate system (latitude and longitude).

- *domainExtent* describes the spatial extent of the domain of the orthoimagery. The constraint [1..*] for this attributes indicates that at least one domainExtent, represented as data type EX_Geographic Extent (ISO 19115) is required, but there can be multiple instances of extent. For instance, geographic extent can be described in terms of a bounding polygon made up of coordinates, a bounding box made up of coordinates, or textual description of the geographic extent, or any combination of these three. The example provides a bounding box described by longitude and latitude:
  - *westBoundLongitude: 76.00000*
  - *southBoundLatitude: 39.46667*
  - *eastBoundlatitude: 75.91667*
  - *northBoundlatitude: 39.50000*

- *rangeType* describes the attribute range of the orthoimage. This attribute uses a special data type called RecordType which appears in the format “attribute name : data type”. For example, images with single bands will have only one value per pixel, and the *rangeType* can be expressed as “grayscale : integer”. Multi-band images such as natural color images will have three or more values per pixel, which is the case for the example:
  - red: Integer
  - green: Integer

    - blue: Integer
- *interpolationType:* resampling algorithms use methods of interpolation used to transform image values to fit orthorectified geometry. Nearest neighbor, bilinear and cubic convolution are three commonly used methods, however nearest neighbor resampling is not because of the disjointed appearance in the output due to spatial offsets as great as one-half pixel. Images transformed using bilinear interpolations are generally acceptable (bilinear is the default value assumed for interpolationType). A precise resampling method such as cubic convolution is recommended.

- *commonPointRule* identifies the procedure recommended for returning a value if the orthoimage is queried, and the location of the query falls on a boundary between two pixels. Three choices are possible: high, average, low. If average is selected, as in the example, the average of the values of the adjacent pixels will be returned.

- *metadata* provides a link to metadata that describes the orthoimage, for instance how it was collected, when it was collected, accuracy testing procedures used, and so forth.

- *Origin* specifies the origin of the image in an external coordinate reference system. The origin of an orthoimage is always the upper-left pixel. This attribute uses a special data type called DirectPosition which has four required attributes, described below. The attributes *kindCode* and *codeSpace* are from the SC_CRS class defined in ISO 19111: Geographic Information – Spatial referencing by coordinates.
    - *coordinate:* A pair of coordinates to describe the location of the center of the origin pixel (the coordinates of the rest of the image can be calculated from this position because of the regular spacing of the grid). The example is in latitude and longitude: 39.500, 76.000
    - *dimension:* always 2 for an orthoimage, because by definition an orthoimage is a two-dimensional grid with two axes (x,y)
    - *kindCode*: The value for any 2D horizontal coordinate reference system is 1, generalCase.
    - *codeSpace*: a code to identify the coordinate system and its datum, e.g. GCS_North_American_1983, which is shortened to NAD83 in the example.

- *offsetVectors* are used to specify the interval between the coordinates of the grid, as shown in Figure 3.2.3, where the offset vectors both in the x and y dimension have a length of 2 meters. This attribute uses a data type called vector, which not only stores information about distance (length) but also direction.
    - *dimension:* always 2, because an orthoimage is a two-dimensional grid with two axes (x,y)
    - *ordinates:* Coordinates that describe the position of one end of a vector when the other end is taken to be at the origin of the coordinate reference system. Since there are two vectors (one for the x axis, one of the y axis) there will be two ordinates. The example ordinates are -0.00028,0 and 0, -0.00028 (units are degrees because the coordinates system is geographic).

- *extent* is defined by the corner of the grid with the lowest grid coordinates values (usually 0,0) and the corner of the grid having the highest grid coordinate values (for example 400,400 are the highest grid coordinate values for a grid with 400 rows and 400 columns). It is important to note that grid coordinate values (CV_GridCoordinate) do not use the external coordinate system used to reference the grid to the real world. The value of a grid coordinate is a pair of integers representing the number of offsets from the origin of the grid in the direction of each axis. The example identifies the lowest grid coordinates as 0,0 and the highest coordinates as 120,3000.

- *startSequence:* since it is not necessary to explicitly define the horizontal geometry coordinate pairs of each pixel or cell in a grid, this means that the orthoimagery values are not explicitly defined by coordinate pairs, either. Instead, the values are referenced to grid geometry by a sequence of records. This attributes indicates the coordinate where the sequence begins. For the example, the coordinate is 0,0.

- *sequencingRule:* ordering of the values starts by incrementing coordinates along one grid axis, beginning at the specified starting coordinate. When it reaches the end of the first axis, it begins to increment coordinates along a second grid axis, then a third, and so on until it has progressed in the direction of each of the grid axes. The *sequencingRule* attribute is made up of a special data type that defines the type of sequencing (e.g. linear) and the scan direction, or the order of axes that the sequences progresses along. Two examples of linear sequencing are shown in Figure A.2 in Annex A of Part 2: Digital Orthoimagery. Linear sequencing is just one type of method; the complete codelist of CV_SequenceType is given in Table A.4 of Part 2: Digital Orthoimagery. The values from the example are:
  - *type*: linear
  - *scanDirection*: column, row. This means the sequence progresses from the origin down the first column. When it reaches the bottom of the column, it moves the next row and proceeds from the top to the bottom of the second column, and so forth.

- *values:* a set of values for each of the attributes listed in rangeType. The number of sets will match the number of pixels in the image. In the example, each set includes three values matching the red, green and blue attributes.
  - 239, 17, 128
  - 37,219,50
  - Etc, for a sequence of 36421 records

## 3.3 Framework Data Content Standard Part 3: Elevation

### 3.3.1 Introduction

Digital elevation data are sets of elevation measurements for locations distributed over the land surface. The Elevation part of the Framework Data Content Standard includes both topographic elevation data (above a reference datum) and bathymetric elevation data (below a reference datum). The reference datum is usually either sea level or the geoid.

Elevation data have many practical uses ranging from environmental to urban. Slope and aspect can be directly derived from elevation. Elevation datasets are also used for scenario analysis ranging from calculations of cut and fill requirements by engineers for projects relating to road construction to viewshed analysis. Viewshed analysis (or line-of-sight) uses topographical data to determine the visibility of areas from a given point.

Elevation data can be generated from existing contour maps, photogrammetric analysis of stereo aerial photography, satellite imagery, GPS, or laser-based (LiDAR).

Digital elevation data can be represented several different ways: points, grid, contours, TIN (triangulated irregular network), and profiles. Each representation was developed for specific uses and has its own set of pros and cons for different type of applications. For instance, an elevation grid is useful for deriving slope and aspect and for combination with other data for analysis, whereas elevation contours are helpful for indicating elevation on a map that has some other primary theme, such as landforms or man-made structures. The difference in models is further complicated by different software formats, and terms used to define models may have subtle differences in meaning due to terminology adopted by proprietary data formats. This part of the Standard provides methods of describing these common elevation data models, so that regardless of the way the elevation data is represented, it can be included as Framework data.

A bibliography is provided in Annex C of Part 3: Elevation. In addition to these sources, the *FGDC online training materials webpage* has materials for download (PowerPoint slides) specific to Part 3: Elevation. These materials include other graphic examples of different types of elevation data, as well as descriptions and links to several websites that describe or serve elevation data.

### 3.3.2 Requirements

Framework requires that elevation be described with one or more of five data models: Point, Grid, TIN, Contour or Profile. The application schemas provided in the annexes of Part 3: Elevation accommodate the level of content and relationships necessary for exchange of all forms and formats of these data in a predictable and repeatable manner.

The requirements in Section 7 of the Part 3: Elevation document provides a detailed description of each of the five data models, along with figures of examples of each. The application schemas for each of the models are provided in Annex A to Part 3, and the data dictionaries are provided in Annex B to Part 3.

If implementers of the elevation part of the Framework Standard need more specifics about each of the data models described in the requirements section of Part 3, they should reference the document Guidelines for Digital Elevation Data, produced by the National Digital Elevation Program (*www.ndep.gov*). This document describes these geospatial data models in more detail and provide best practices and examples for each.

This part of the guidance document should be used in conjunction with Annex A and B to help clarify the elevation collection diagram (subchapter 3.3.2.1), the class diagram for the abstract type ElevationCoverage (subchapter 3.3.2.2), and the specific class diagrams for each of the elevation data models (subchapter 3.3.2.3).

The following subchapters include definitions and explanations of the following terms:

- Aggregation
- Coverage
- Domain (with respect to the coverage type)
- Range (with respect to the coverage type)

The following subchapters refer to terms defined in other subchapters: *abstraction, instance (instantiation), generalization, conceptual schema, application schema (subchapter 2.2), UML classes and stereotypes (subchapter 2.3.1.2), attributes, multiplicity, data type (subchapter 2.3.1.3).*

### *3.3.2.1 ElevationCollection*

The class ElevationCollection (see Figure 3.3.1) represents a collection of elevation data. For instance, this may be a collection associated with a data transfer request. ElevationCollection is an instantiation of the FeatureCollection class specified in Part 0: Base document and inherits the attribute, metadata, from FeatureCollection.

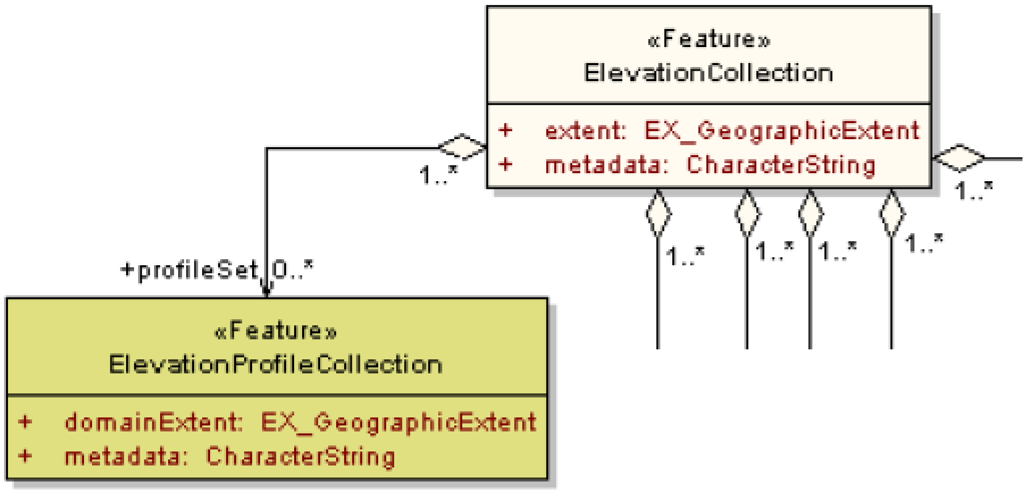

«Feature»
ElevationCollection
+ extent: EX_GeographicExtent
+ metadata: CharacterString
1..*
1..*
1..*
1..*
1..*
1..*
+profileSet
0..*
«Feature»
ElevationProfileCollection
+ domainExtent: EX_GeographicExtent
+ metadata: CharacterString

**Figure 3.3.1**
**ElevationCollection class (detail of figure A.1 in Annex A of Part 3: Elevation)**

A collection must include at least one of six <<Feature>> classes specific to elevation. In Figure 3.3.1, only one of the six, ElevationProfileCollection, is shown, for simplicity. The other five are represented in this class diagram the same way as ElevationProfileCollection. They are ElevationGridCoverage, ElevationTinCoverage, ElevationPointCoverage, ElevationContourCoverage and ElevationPointCollection.

An ElevationCollection may contain any combination of any number of these elevation features. Each <<Feature>> class in Figure A.1 in Annex A of Part 3 is described in further detail with its own UML class diagram and corresponding data dictionary table.

In addition to the attribute *metadata*, ElevatonCollection must also have an attribute, *extent*. Extent describes the geographic area of the ElevationCollection, which encompasses the geographic extents of all the instances of elevation within the collection. It is possible that the elevation within the collection may not be geographically contiguous. A data request for Framework data may include two completely different, non-related areas of interest. The data type EX_GeographicExtent (described in ISO 19115: Geographic Information: Metadata) must be used to describe the extent.

#### *3.3.2.2 ElevationCoverage*

Four of the six elevation feature types are subtypes of ElevationCoverage. The abstract **Coverage** type is a way of describing geographic information that is continuous in nature, rather than discrete in nature. Discrete phenomena are recognizable objects that have relatively well-defined boundaries or spatial extent. Examples include buildings, streams and measurement stations. Continuous phenomena vary over space and have no specific extent. Examples include temperature, soil composition and elevation. A value or description of a continuous phenomenon is only meaningful at a particular position in space (and possibly time). Coverage refers to any data representation that assigns values directly to spatial position. More specifically, a coverage associates positions within a bounded space (its **domain**) to attribute values (its **range**). For instance, within the domain of a collection of points representing locations across the state of California, elevation has a range of -282 ft below sea level to 14505 feet above sea level. For any direct position (pair of coordinates) within this domain, an elevation coverage will return a value within the range of -282 to 14505. So in other words, a coverage is both a feature and a function (OGC, 2006).

Coverage geometry refers to the configuration of the domain of a coverage in terms of coordinates. This part of the Framework Standard allows four types of coverages that reflect different arrangements of coverage geometry: ElevationGridCoverage, ElevationTinCoverage, ElevationProfileCoverage and ElevationPointCoverage.

An issue pertaining to continuous data is that it is difficult and expensive to collect. True continuous collection requires literally covering every inch of the earth's surface. Aerial photographs, satellites, and LiDAR sensors are able to do this, but they are also limited in terms of resolution and other factors related to the collecting sensors. Therefore, many continuous coverages are actually generated from discrete objects, for instance sample locations. The continuous coverage is usually created by interpolating elevation from between locations of known elevation. Most interpolation algorithms depend upon a structured pattern of spatial relationships between the points. This requires either that the points be arranged in a regular way, such as a grid, or that the spatial domain of the continuous coverage be partitioned in a regular way in relation to the points, such as a triangulated irregular network (TIN). The TIN coverage involves interpolation of values within triangles composed of three neighboring points which represent known or calculated elevation. Known elevation may be collected a number of ways: traditional surveying methods, GPS, LiDAR. Elevation can also be calculated from contours, or from aerial photographs via photogrammetry methods.

ElevationContourCoverage and ElevationPointCoverage represent cases where the elevation is represented as discrete coverages rather than continuous coverages. Contours are discrete objects but are treated as subtypes of a coverage because they still represent elevation continuously in a visual manner. Points are a coverage subtype because they are used as the basis for interpolation of a continuous coverage. For instance, a data manager can use elevation points to create a grid or TIN coverage but may also want to maintain the points as a separate dataset for quality control or other purposes.

The ElevationCoverage class as four required attributes, as shown in Figure 3.3.2.

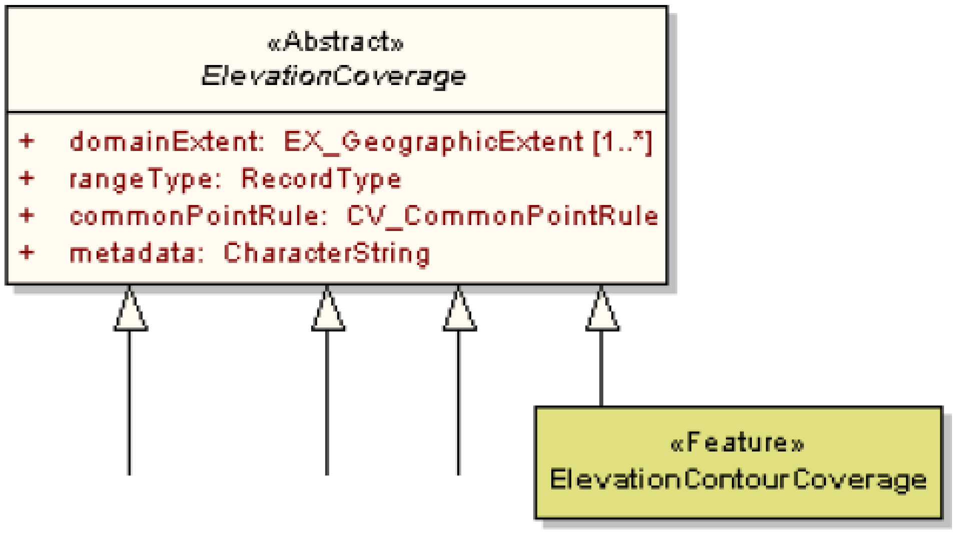

«Abstract»
ElevationCoverage
+ domainExtent: EX_GeographicExtent [1..*]
+ rangeType: RecordType
+ commonPointRule: CV_CommonPointRule
+ metadata: CharacterString
«Feature»
ElevationContourCoverage

**Figure 3.3.2**
**ElevationCoverage class (detail of figure A.2 in Annex A of Part 3: Elevation)**

Since ElevationCoverage is an abstract class, it must be instantiated as one of four subtypes (Figure 3.3.2 shows only one of the four, the sake of simplicity since they are all represented with the same relationship as ElevationContourCoverage). Each of the four subtypes will inherit the four attributes, described below:

- *domainExtent* describes the spatial extent of the domain of the ElevationCoverage. The constraint [1..*] for this attributes indicates that at least one domainExtent, represented as data type EX_GeographicExtent (ISO 19115) is required, but there can be multiple instances of extent. For instance, geographic extent can be described in terms of a bounding polygon made up of coordinates, a bounding box made up of coordinates, or textual description of the geographic extent, or any combination of these three.
- *rangeType:* describes the attribute range of the ElevationCoverage. It uses the <<DataType>> RecordType specified in ISO/TS 19103.
- *commonPointRule* identifies the procedure recommended for evaluating the ElevationCoverage at a position that falls on a boundary between geometric objects in the domain of the coverage.

- *metadata* provides a link to metadata that describes the ElevationCoverage, for instance how it was collected, when it was collected, quality checking procedures used, and so forth.

**FAQ:** How is the range described for a coverage?

An elevation coverage can include values for different types of elevation. For example, reflective surface elevation (the first reflective surface a sensor detects, such as treetops, rooftops, and other natural or manmade features), bare earth surface, and bathymetric surface (surface of the ground under water). In order to accommodate these three different values, the rangeType attribute uses a special data type, <<RecordType>>, to produce the following list, in the format of name: data type.

reflective surface elevation: Real
bare earth surface elevation: Real
bathymetric surface elevation: Real

**FAQ:** What is the coverage attribute *commonPointRule* used for?

An ElevationCoverage is both a feature and a function. As a function, for any direct position (pair of coordinates) within its domain, an elevation coverage will return a value within its range. Whichever rule specified by commonPointRule (average, low, high) shall be applied to the set of elevation values that results from each of the geometric objects that share a boundary. For instance, if the direct position within an ElevationTinCoverage is exactly at the boundary between two triangles in a TIN, then the coverage would return an elevation value that is either the average, low or high of the two values of the geometric objects.

*commonPointRule* gets its values from the code list CV_CommonPointRule specified in ISO 19123. For elevation coverages, appropriate values of CV_CommonPointRule include "average", "high", and "low".

average – return the mean of the feature attribute values
low – use the least of the feature attribute values
high – use the greatest of the feature attribute values

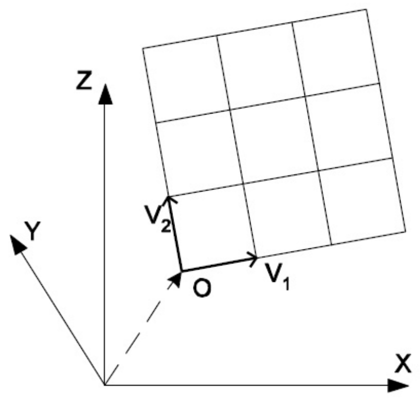
#### *3.3.2.3 ElevationGridCoverage*

This subchapter references Figure A.3, the UML class diagram for ElevationGridCoverage, and Table B.3, Data Dictionary for ElevationGridCoverage, found in the annexes of the Framework Standard for Elevation (Part 3). All of the attributes listed in the data dictionary table are mandatory. This subchapter describes each of these attributes in more detail. The ElevationGridCoverage class is a realization of (uses the same attributes of) the CV_ContinuousQuadrilateralGridCoverage in the conceptual schema for the coverage (ISO 19123). It also inherits attributes from ElevationCoverage.

A grid is probably the most common way of representing elevation, because the structure is very efficient. The regular spacing of a grid requires that only one point be referenced to a horizontal coordinate. From this grid origin, the horizontal location of all other points can be determined. This eliminates the need to explicitly define the horizontal geometry coordinate pairs of each elevation and minimizes file size. The grid is also an efficient structure for data processing because of its regular nature.

An ElevationGrid is usually created from a regular spacing of points with known elevation. From these discrete points, the continuous values for the grid are calculated using interpolation. Interpolation methods are also used to fit grid coordinates to a geographic or projected coordinate reference system such as latitude/longitude or Universal Transverse Mercator. For an ElevationGrid, the *interpolationType* attribute must be either "bilinear" or "bicubic". Table A.2 in Part 3: Elevation gives the complete codelist of interpolation methods; for instance, the continuous coverage of a triangulated irregular network (TIN) uses a different *interpolationType*.

Figure 3.3.3 is provided to help visualize four attributes of a grid: *dimension, axisNames, origin*, and *offsetVectors*. These attributes are inherited from CV_RectifiedGrid (ISO 19123).



**Key**

| | |
|---|---|
| X, Y, Z | axes to determine 3-space |
| $V_1$, $V_2$ | offset vectors |
| 0 | grid origin |

**Figure 3.3.3**
**Geometry of a Rectified Grid (reproduced from Figure14 in The OpenGIS Abstract Specification Topic 6: Schema for coverage geometry and functions).**

It should be noted that Figure 3.3.3 has three axes (x,y,z) whereas an ElevationGrid is limited to two axes (and therefore, two dimensions). One common source of elevation, the USGS DEM, refers to the *axisNames* as "row" and "column".

The attribute *origin* specifies the origin of the rectified grid in an external coordinate reference system; in other words, the intersection of the x and y axes (see Figure 3.3.3). For example, if the ElevationGrid is in a geographic coordinate system, the origin would be identified by a latitude, longitude, and a horizontal datum (the vertical datum used for elevation values should be recorded in the metadata). The origin is described using a data type called direct position, defined in ISO 19111 for describing coordinate reference systems. An example is given in Table 3.3.1.

The *offsetVectors* are used to specify the interval between the coordinates of the grid, as shown in Figure 3.3.3. This attribute uses ordinates to describe the distance and direction of the vectors. The beginning of a vector is the origin of the coordinate reference system, and a positive or negative number indicates the distance and direction from the origin. Since an ElevationGrid must have only two dimensions, there are two vectors (one for the x axis, one of the y axis) and therefore two ordinates. Example ordinates are 30,0 and 0,30 for an ElevationGrid with 30 meter cells.

The attribute *extent* provides area of the grid, and uses CV_GridEnvelope (ISO 19123) as its data type. The extent is defined by the corner of the grid with the lowest grid coordinates values (usually 0,0) and the corner of the grid having the highest grid coordinate values (for example 400,400 are the highest grid coordinate values for a grid with 400 rows and 400 columns). It is important to note that grid coordinate values (CV_GridCoordinate) do not use the external coordinate system used to reference the grid to the real world. The value of a grid coordinate is a pair of integers representing the number of offsets from the origin of the grid in the direction of each axis. Another example is given in Table 3.2.1.

Since it is not necessary to explicitly define the horizontal geometry coordinate pairs of each elevation in a grid, this means that the elevation values are not explicitly defined by coordinate pairs, either. Instead, the elevation values are referenced to grid geometry by a sequence of records. In order to reference the values to the grid geometry, three attributes are required: *sequencingRule, startSequence,* and *values*, which come from the CV_GridValuesMatrix class (ISO 19123).

Ordering of the grid cells starts by incrementing coordinates along one grid axis, beginning at the specified starting coordinate (*startSequence*). When it reaches the end of the first axis, it begins to increment coordinates along a second grid axis, then a third, and so on until it has progressed in the direction of each of the grid axes. The *sequencingRule* attribute is made up of a special data type that defines the type of sequencing and the order of axes. Several examples of sequencing are shown in Figure 3.3.4. One common source of elevation, the USGS DEM, is a plain ASCII file where the elevation values stored in the file represent profiles (columns of a grid) that are linearly sequenced in the (y,x) order shown in Figure 3.3.4.

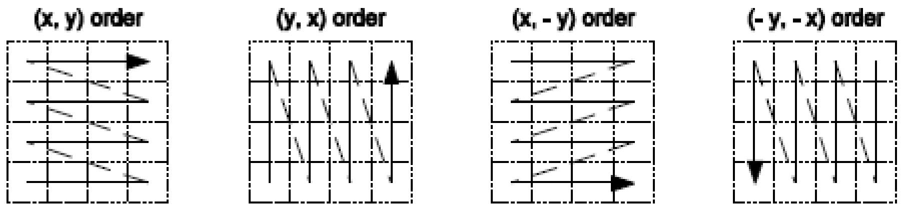


**Figure 3.3.4**
**Four different examples of linear sequencing of values associated with a grid.**

Linear sequencing is just one type of method; the complete codelist of sequencing methods available is given in Table A.3 of Part 3: Elevation.

A data example that has all the content required for an ElevationGridCoverage is provided in Table 3.3.1.

**Table 3.3.1**
**Mandatory attributes for an ElevationGridCoverage and example values.**

| Name | Value | | |
|---|---|---|---|
| ElevationCoverage | | | |
| domainExtent | *westBoundLongitude* | -107.771 | |
| | *southBoundLatitude* | 41.309 | |
| | *eastBoundLongitude* | -107.266 | |
| | *northBoundLatitude* | 41.588 | |
| rangeType | bare surface elevation: real | | |
| | first reflective surface: real | | |
| interpolationType | bilinear | | |
| commonPointRule | average | | |
| ElevationGrid | | | |
| Dimension | 2 | | |
| axisNames | row, column | | |
| Origin | *coordinate* | 268394.19, 4576841.51 | |
| | *dimension* | 2 | |
| | *coordinateReferenceSystem.kindcode* | Compound | |
| | *coordinateReferenceSystem.name* | NAD_1983_UTM_Zone_13N | |
| offsetVectors | *dimension* | *Ordinates(1)* | *Ordinates(2)* |
| | 2 | 9.1650012 | 0 |
| | 2 | 0 | 9.1650012 |
| Extent | *low* | 0,0 | |
| | *high* | 4533, 3347 | |
| startSequence | | 0,0 | |
| sequencingRule | *type* | Linear | |
| | *scanDirection* | row, column | |
| Values | 2067,2067 | | |
| | 2454,2460 | | |
| | etc., set of values for each attributes listed in rangeType, for a total of x records | | |

#### *3.3.3.4 ElevationPointCoverage*

This subchapter references Figure A.4, the UML class diagram for ElevationPointCoverage, and Table B.4, Data dictionary for ElevationPointCoverage, found in the annexes of Part 3: Elevation.

An ElevationPointCoverage is an aggregation of points, each of which is associated with one or more elevation values carried as attributes (rather than including the elevation in the set of coordinates, e.g. x,y,z).

The class ElevationPointCoverage is a different type of coverage than the continuous coverage used to represent the ElevationGrid in the previous subchapter.  It is based on CV_DiscretePointCoverage, specified in ISO 19123. A discrete point coverage is characterized by a finite domain consisting of points, usually irregularly distributed. The principal use of discrete point coverages is to provide a basis for continuous coverage functions, where the coverage will return a value for any direct position within the coverage by interpolation between the points of the discrete point coverage. This data model is provided in the Framework Standard for Elevation because there may be cases where the original data points used to create either an elevation grid or TIN should be preserved separately from the continuous model. For instance, by adding new samples to an ElevationPointCoverage over time or by correcting the elevations associated with existing points, more accurate grids and TINS can then be generated from the ElevationPointCoverage.

An ElevationPointCoverage is a collection of MultiElevationPoints: each point has a x,y location (attribute *geometry*, data type GM_Point from ISO 19107) that has a record of one or more elevation values associated with it (attribute *value,* data type Record). As discussed in subchapter 3.3.3.2, there may be more than one elevation associated with a Record: such as a bare earth elevation, a reflected surface elevation (such as a treetop), or a bathymetric (under the surface of water) elevation. These are identified by the rangeType attribute inherited from ElevationCoverage.

#### *3.3.3.5 ElevationTinCoverage*

This subchapter references Figure A.5, the UML class diagram for ElevationTinCoverage, and Table B.5, Data dictionary for ElevationTinCoverage, found in the annexes Part 3: Elevation.

A triangulated irregular network (TIN) is a surface representation derived from irregularly spaced sample points and breakline features (surface discontinuities such as peaks, pits, ridges, and valleys). Each sample point has an x,y coordinate geometry pair and a surface, or z-value. These points are connected by edges to form a set of non-overlapping triangles used to represent the surface. TINs allow for extra data in complex areas and less data in non-complex areas and enable the use of natural topographic features as breaklines.

The class ElevationTinCoverage is a realization of the Type CV_TinCoverage specified in ISO 19123. It is an aggregation of ElevationTriangles that represents a triangulated irregular network in which the points of known elevation fall on the vertices of the triangles. It is also a subclass of ElevationCoverage that inherits the attributes specified for that class.

The attribute *interpolationType* specifies the interpolation method recommended for interpolating elevation from the three points that make up each triangle. Barycentric is the recommended method for TINs, so this is the required value for this attribute.

The attribute *geometry* represents the network of triangles that form the basis of the TIN and uses data type GM_Tin from ISO 19107. The triangles must lie on a 2-dimensional surface; the elevation values at the vertices are treated as attributes of the points, not as coordinate values such as x,y,z.

An ElevationTinCoverage is a collection of ElevationTriangles. Each ElevationTriangle is a collection of exactly three MultiElevationPoints: each point has an x,y location (attribute *geometry*, data type GM_Point from ISO 19107) and a record of one or more elevation values associated with it (attribute *value,* data type Record). As discussed in subchapter 3.3.3.2, there may be more than one elevation associated with a Record: such as a bare earth elevation, a reflected surface elevation (such as a treetop), or a bathymetric (under the surface of water) elevation. These are identified by the rangeType attribute inherited from ElevationCoverage.

#### *3.3.3.6 ElevationContourCoverage*

This subchapter references Figure A.6, the UML class diagram for ElevationPointCoverage, and Table B.6, Data dictionary for ElevationPointCoverage, found in the annexes of Part 3: Elevation.

Contours are vectors connecting points of equal elevation and are a common visual representation of topography and bathymetry in mapping applications. The class ElevationContourCoverage is a realization of the Type CV_DiscreteCoverage specified in ISO 19123 that represents a set of elevation contours. It is also a subclass of ElevationCoverage that inherits the attributes specified for that class.

An ElevationContourCoverage is a collection of one or more ElevationContours. The ElevationContour class has an attribute *geometry* that uses data type GM_Curve from ISO 19107. Each ElevationContour made up of two or more x,y coordinate pairs defining the length and shape of the contour. Each contour has one elevation value associated with it, stored as a number in the *value* attribute.

#### *3.3.3.7 ElevationPointSet*

This subchapter references Figure A.7, the UML class diagram for ElevationPointSet, and Table B.7, Data dictionary for ElevationPointSet, found in the annexes of Part 3: Elevation.

The class ElevationPointSet represents a collection of points each related to a 3D coordinate reference system. The elevation value is carried as one of the coordinates (x,y,z) rather than as a distinct attribute of the point.

The class ElevationPoint represents a point associated with a single elevation surface. Unlike the MultiElevationPoints of an ElevationPointCoverage, each ElevationPoint in an ElevationPointSet has only one elevation value and represents only one elevation surface type. The attribute *surfaceType* identifies the type of surface that is described by the ElevationPoint (e.g. bare ground surface, first reflective surface).

The attribute *geometry* uses an instance of GM_Point from ISO 19107. The position of the point is described in terms of a 3D coordinate reference system (x,y,z) where one of the coordinates is the value for the elevation. The units of the elevation should be provided in the metadata.

#### *3.3.3.8 ElevationProfileCollection*

This subchapter references Figure A.8, the UML class diagram for ElevationProfileCollection, and Table B.8, Data dictionary for ElevationProfileCollection, found in the annexes of the Framework Standard for Elevation (Part 3).

While a contour connects points of equal elevation, a profile connects points of varying elevation. Profiles are commonly used to model surface discontinuities such as peaks, pits, ridges and valleys. They are also used to model the elevation changes in man-made linear features such as roads or pipelines, or cross sections of elevation that lie perpendicular to a linear feature.

The class ElevationProfileCollection is a collection of ElevationProfiles. ElevationProfile uses the data type GM_Curve, specified in ISO 19107. In this particular instance, the GM_Curve must be made up of two or more x,y,z coordinate sets where the x,y coordinates define elevation locations along the profile and the z coordinate provides the elevation value of each location.

The attribute *surfaceType* identifies the type of surface that the elevation values describe (e.g. ground surface, first reflective surface). The attribute *profileType* identifies the kind of profile from the code list ProfileType (Table A.17.4 from Part 3: Elevation). ProfileTypes include transect, breakline, traverse, ridgeline, or drainageline.

## 3.4 Framework Data Content Standard Part 4: Geodetic Control

### 3.4.1 Introduction

Geodesy, also called geodetics, is the science concerned with determining the size and shape of the Earth and the location of points upon its surface. Geodetic control refers to locations on the earth's surface, usually identified by some sort of monument or marker, whose coordinates provides a common reference system for establishing coordinates for all other geographic data.  Geodetic control points are a type of control point.

Control points have several common characteristics:

- They are physical points on the ground which can be revisited or located for future use
- They are used for subsequent projects, that is to say, they themselves are not the end product
- Their coordinates are determined using more accurate techniques because they will be used to control or fit future spatial data activities

The definition for geodetic control provided in this part of the Framework Standard is "a set of control points whose coordinates are established by geodetic surveying methods" such as classical line-of sight triangulation, traverse, and geodetic leveling; or satellite surveys such as Doppler or GPS.

Geodetic control information plays a crucial role in developing all Framework data and users' applications data, because it provides the spatial reference source to register all other spatial data. In addition geodetic control information may be used to plan surveys, assess data quality, plan data collection and conversion, and fit new areas of data into existing coverages (FGDC, 1997). Geodetic Control supports accurate horizontal and vertical placement of all other layers, particularly cadastral (by improving horizontal locations), elevation (by providing accurate elevations), and by providing a means for orthorectifying aerial photos into orthoimagery.

In the United States, the National Geodetic Survey (*www.ngs.noaa.gov*) is responsible for maintaining the National Spatial Reference System (NSRS), the fundamental geodetic control for the United States. Section 1.1 of Part 4: Geodetic Control provides more information about geodetic control surveys and the NSRS.

It is important to note that the scope of the Geodetic Control part of the Framework Data Content Standard can go beyond control points that are part of the NSRS. Many government agencies and industries create geodetic control points for their own uses, where the networks in the NSRS are not dense enough. For instance, Minnesota's Geodetic Unit, in cooperation with other state agencies such as the Department of Transportation, are working to place permanent physical monuments at three-mile intervals across Minnesota. These densification projects are helping to develop a statewide geodetic control network with centimeter accuracy. This accuracy provides better geodetic control for right-of-way acquisition, construction, maintenance, and GIS data collection.[2]

[2] *http://www.gis.state.mn.us/MSDI/workgroups/geodetic.htm*, accessed May, 2010

While geodetic control are points whose coordinates have been obtained by precise geodetic surveying methods, there are several other types of control points which may also be represented using the Geodetic Control part of the Framework Standard. These include Public Land Survey System (PLSS) points, property corners, photo control points, right-of-way points and other types of local points whose coordinates have been determined by methods perhaps not as rigorous as geodetic surveying methods. For more information about using Part 4: Geodetic Control for these types of control points, see Annex D of Part 4.

**FAQ:** Does geodetic control data seeking to meeting Framework Data Content Standards have to meet the same rigorous requirements as NSRS control stations?

No, but it is recommended. State and local governments and industry regularly undertake geodetic control surveys for their specific surveying or mapping needs. These types of geodetic control supplement the NSRS, and depending on their level of accuracy may be submitted to NGS for inclusion into the NSRS. However, the main purpose of this part of the Framework Data Content Standard is to support the exchange of geodetic control data, in particular by establishing a common baseline for the semantic content geodetic control databases. While NSRS has specific requirements for accuracy, content and format, this part of the Framework Standard determines a standard way of describing geodetic control points, regardless of their accuracy or format. The application schema or logical model provided for geodetic control allows for transformation of geodetic control databases (in any format) into a standardized implementation which can be readily combined with other sources of geodetic control data.

A data example for a sample geodetic control point is provided in Annex B of Part 4: Geodetic Control. A bibliography is provided in Annex E. In addition to these sources, the *FGDC online training materials webpage* has materials for download (PowerPoint slides) specific to Part 4: Geodetic Control. These materials include other graphic examples of different types of geodetic controls, as well as descriptions and links to several websites that describe or serve geodetic control data.

### 3.4.2 Requirements

Every geodetic control point shall have four basic elements:

- Designations (unique identifier, link to external resources)
- Coordinates (horizontal and vertical)
- Accuracy
- Geodetic Datum

Because the geodetic datum is essential to describing the coordinates and accuracy of geodetic control points, the order of the following subchapters has been changed to discuss geodetic datum before coordinates and accuracy.

#### *3.4.2.1 Designations*

Designations refer to three types of identifiers used for each point in the dataset:

- Unique identifier (mandatory)

- descriptive identifier (optional)
- a Uniform Resource Identifier (URI) (optional).

A unique identifier for each point within a dataset shall be composed of two parts: 1) a permanent identifier and 2) a namespace. The permanent identifier can be the organization's unique database identifier. For example, unique = MN0298. The namespace is the organization's identifier (for example, abbreviation) for the organization who assigned/maintains the permanent identifier. For example, uniqueIDAssigner = NGS.

The unique identifier allows traceability of each data point back to the organization and to other data held by that organization about the point. For example, NGS has a multitude of information about each geodetic control point, but only the basic information conforming to this part need be contained in the produced dataset.

For geodetic control datasets, the uniqueness of namespace is maintained by the National Geodetic Survey through Input Format and Specifications of the National Geodetic Survey Data Base, Appendix C - Contributors of Geodetic Control Data, FGCS, 1994. For more details about the unique identifier, see section 6.2.1 in Part 4: Geodetic Control.

A descriptive identifier is optional, but a meaningful name such as for a landmark near the point, can help with locating the monument or marker in the field. Descriptive identifiers do not have to be unique within a dataset.

A permanent URI, such as a URL, can provide the user with a direct link to an Internet-based resource that facilitates certain interactions with the point, for example, a link to an NGS datasheet or a scanned tie-sheet image. URI do not have to be unique within a dataset.

#### *3.4.2.2 Geodetic datum*

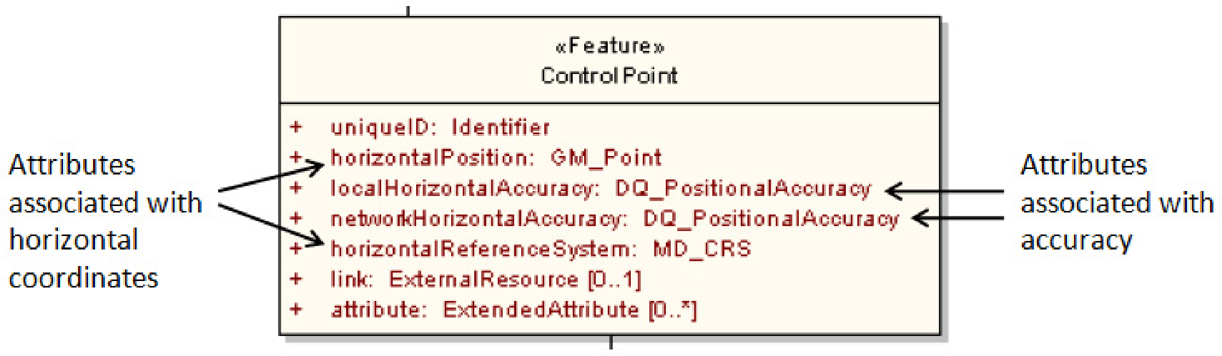
A geodetic datum is a set of reference points on the Earth's surface against which position measurements are made, and are associated with a model of the shape of the earth (ellipsoid or geoid). Horizontal datums are used for describing a point on the earth's surface. Vertical datums measure elevations or depths.

In this part of the Framework Standard, horizontal coordinates and ellipsoid heights shall be referenced to the North American Datum of 1983 (NAD 83). NAD83 is based on the adjustment of a horizontal control network of geodetic control points to the Global Reference System of 1980, a very accurate model of the shape of the earth. The horizontal control network consists of over 250,000 monumented control stations that are spread across the continental United States and interconnected by surveying observations.

References to NAD83 shall include the datum tag and the coordinate epoch date. The datum tag (for example "NAD 83 (1986)") represents the date of the regional least squares adjustment associated with the horizontal geodetic control point. The epoch date (for example 2003.0) shall be used for control points in regions of episodic and/or continuous horizontal and vertical crustal motion where the coordinates change with time. For more details about the datum tag and epoch date, see section 6.5 in Part 4: Geodetic Control.

### 3.4.2.3 Horizontal coordinates

Horizontal coordinates provide the location on the ground, and must be recorded in latitude and longitude, and the unit must be decimal degrees. Latitudes shall be referenced as positive north and negative south. Longitudes shall be referenced as positive east and negative west.



«Feature»
ControlPoint
+ uniqueID: Identifier
+ horizontalPosition: GM_Point
+ localHorizontalAccuracy: DQ_PositionalAccuracy
+ networkHorizontalAccuracy: DQ_PositionalAccuracy
+ horizontalReferenceSystem: MD_CRS
+ link: ExternalResource [0..1]
+ attribute: ExtendedAttribute [0..*]
Attributes associated with horizontal coordinates
Attributes associated with accuracy

**Figure 3.4.1**
**The ControlPoint class (detail of Figure 2 from Part 4: Geodetic control)**

Several attributes are required for reporting height, as shown in the class for ControlPoint (Figure 3.4.1).

- *hortizontalPosition*: uses the data type, GM_Point, to provide the location of the point as a pair of coordinates.
- *localHorizontalAccuracy: see subchapter 3.4.2.4*
- *networkHorizontalAccuracy: see subchapter 3.4.2.4*
- *horizontalReferenceSystem*: uses the data type, MD_CRS, to provide the following information:
    - *datum*: North American Datum of 1983 is required, and this attribute should also include the datum tag and epoch date (see subchapter 3.4.2.2)
    - *ellipsoid*: the ellipsoid used by NAD83 is GRS 1980
    - *projection*: not used since latitude and longitude is required

### 3.4.2.4 Vertical coordinates

Vertical coordinates consist of two types, orthometric height and ellipsoid height. Orthometric height is measured from the geoid model of the earth's surface; ellipsoid height is measured from the ellipsoid model of the earth's surface. Figure 1 in Part 4: Geodetic Control provides a graphical representation of these different heights.

Either orthometric or ellipsoid height must be provided, and both shall be provided if both are measured.

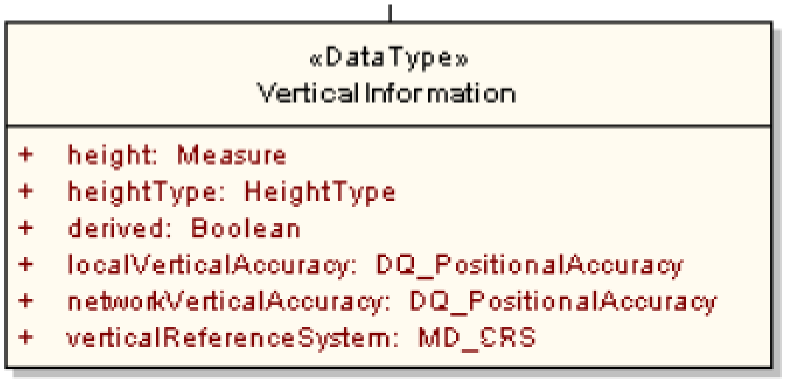

«DataType»
Vertical Information
+ height: Measure
+ heightType: HeightType
+ derived: Boolean
+ localVerticalAccuracy: DQ_PositionalAccuracy
+ networkVerticalAccuracy: DQ_PositionalAccuracy
+ verticalReferenceSystem: MD_CRS

**Figure 3.4.2**
**The VerticalInformation class (detail of Figure 2 from Part 4: Geodetic control)**

Several attributes are required for reporting height, as shown in the class for VerticalInformation (Figure 3.4.2). If reporting both orthometric and ellipsoidal height, all of these attributes must be supplied twice.

- *height*: distance above or below the vertical datum
- *heightType*: Indicator if height is ellipsoidal or orthometric (although this information is inferred from the datum).
- *derived*: Indicator if the vertical information is derived (true) or referenced (false)
- *localVerticalAccuracy*: see subchapter 3.4.2.4
- *networkVerticalAccuracy*: see subchapter 3.4.2.4
- *verticalReferenceSystem*: uses the data type, MD_CRS, to provide the following information:
  - *datum*: Orthometric heights shall be referenced to the North American Vertical Datum of 1988. Ellipsoidal heights shall be referenced to the North American Datum of 1983.

#### *3.4.2.4 Accuracy*

Local and network accuracies shall be provided in meters, expressed at the 95% confidence level. See FGDC-STD-007.2, Geospatial Positioning Accuracy Standards, Part 2: Geodetic Control Networks, for the methodology for defining local and network accuracies.

Local accuracy: correctness of the coordinates of a control point relative to the coordinates of other directly connected , adjacent control points. The reported local accuracy is an approximate average of the individual local accuracy values between this control point and other observed control points used to establish the coordinates of the control point.

Network accuracy: correctness of the coordinates of a control point with respect to the geodetic datum For NSRS network accuracy classification, the datum is considered to be best expressed by the geodetic values at the Continuously Operating Reference Stations (CORS) supported by the National Geodetic Survey (NGS). By this definition, the local and network accuracies at CORS sites are considered to be infinitesimal, that is to say, to approach zero.

Examples:

coordinates – horizontal – accuracy – local = 0.046
coordinates – horizontal – accuracy – network = 0.066

coordinates – vertical – orthometricHeight – accuracy – local = 0.002
coordinates – vertical – orthometricHeight – accuracy – network = 0.100

coordinates – vertical – ellipsoidHeight – accuracy – local = 0.064
coordinates – vertical – ellipsoidHeight – accuracy – network = 0.127

The data type, DQ_PositionalAccuracy that is used to describe local and network accuracy also includes many optional attributes, such as the name and description of the measurement, evaluation method and procedure, and date it was performed. For more details, see ISO 19115 Data Quality section.

See Annex C of Part 4: Geodetic Control for guidance on estimating local and network accuracy values for geodetic control established using the older (for example, first order) methodology.

## 3.5 Framework Data Content Standard Part 5: Governmental Units

### 3.5.1 Introduction

A governmental unit is a geographic area with legally defined boundaries established under Federal, Tribal, State, or local law, and with the authority to elect or appoint officials and raise revenues through taxes. Examples are State, County, American Indian Reservation, City, School District, Village.

This part also describes other types of geographic areas, such as administrative units and statistical areas.

- An administrative unit is a geographic area established by rule or regulation of a legislative, executive, or judicial governmental authority, a non-profit organization, or private industry for the execution of some function. Examples are American Indian Trust Land, Congressional District, Enterprise Zone/Empowerment Community, State Legislative District.
- A statistical unit is a geographic area defined for the collection, tabulation, and/or publication of demographic, and/or other statistical data. Examples are Census Block, Metropolitan Area, Rural-Urban Commuting Area, Unorganized Territory, ZIP Code Tabulation Area.
- Other units are those geographic areas that are not governmental units, administrative units, or statistical units as defined herein, and are not areas that are defined or described in other Framework parts.

The required data content for all these units includes complete identification of the area or boundary and its type. Extensive codelists are provided, along with descriptions, in order to identify standard types.

Optional data may be provided to describe relationships between two or more of these units, for instance if they are adjacent or overlapping, and if there is any dependency between them for maintaining common boundary or area information.

A data example for a sample geodetic control point is provided in Annex A of Part 5: Governmental Units. The *FGDC online training materials webpage* has materials for download (PowerPoint slides) specific to Part 5: Governmental Units. These materials include other graphic examples of different types of units, as well as descriptions and links to several websites that describe or serve governmental, administrative or statistical unit data.

### 3.5.1 Requirements (Content Model)

Governmental units and other units must be described, in terms of their location and extent as a geographic area(s) using a polygon geometry. Optionally, they can also be described as boundaries (curve geometries that are identical to segments of the polygon).

Figure 3.5.1 shows the primary classes in the Governmental Units model.

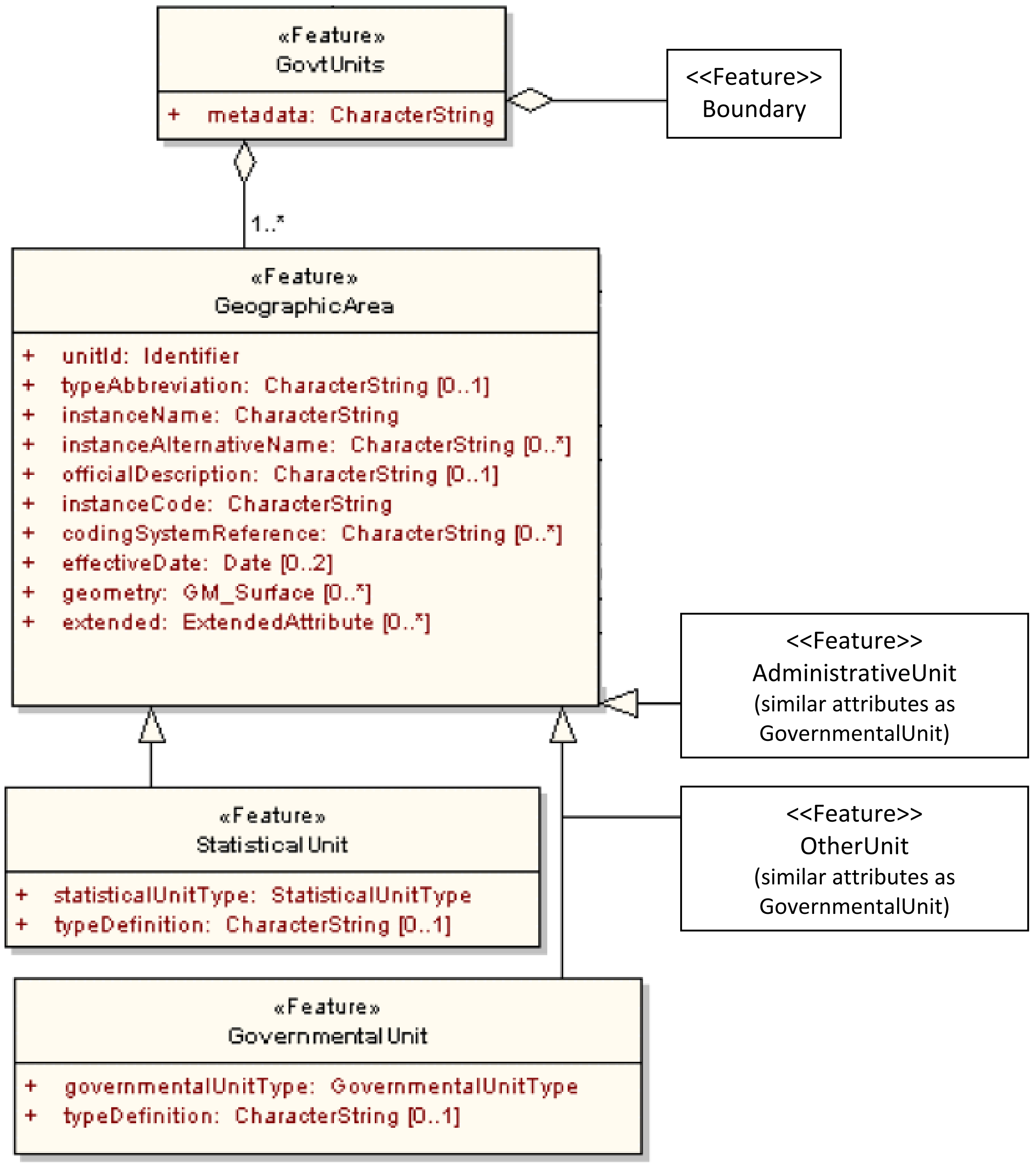

«Feature»
GovtUnits
+ metadata: CharacterString
<<Feature>>
Boundary
1..*
«Feature»
GeographicArea
+ unitId: Identifier
+ typeAbbreviation: CharacterString [0..1]
+ instanceName: CharacterString
+ instanceAlternativeName: CharacterString [0..*]
+ officialDescription: CharacterString [0..1]
+ instanceCode: CharacterString
+ codingSystemReference: CharacterString [0..*]
+ effectiveDate: Date [0..2]
+ geometry: GM_Surface [0..*]
+ extended: ExtendedAttribute [0..*]
<<Feature>>
AdministrativeUnit
(similar attributes as GovernmentalUnit)
<<Feature>>
OtherUnit
(similar attributes as GovernmentalUnit)
«Feature»
Statistical Unit
+ statisticalUnitType: StatisticalUnitType
+ typeDefinition: CharacterString [0..1]
«Feature»
Governmental Unit
+ governmentalUnitType: GovernmentalUnitType
+ typeDefinition: CharacterString [0..1]

**Figure 3.5.1**
**Primary classes of Governmental Units data model (detail from Figure 1 in Part 5: Governmental Units)**

The <<Feature>> GovUnits at the top of Figure 3.5.1 is an instance of FeatureCollection, from the overall Framework application schema specified in Part 0: Base. It represents a collection of governmental unit or other unit features requested for data transfer along with metadata describing the date and content of the transfer.

The <<Feature>> GeographicArea is a parent class that provides attributes to the four subtypes: GovernmentalUnit, StatisticalUnit, AdministrativeUnit and OtherUnit.

The complete UML model is provided in Figure 1 of Part 5: Governmental Units, along with classes to describe spatial relationships and maintenance relationships, which are covered in later sections. They are not included in Figure 3.5.1 in order to provide a simplified view of the model focusing on its required elements.

This subchapter describes the required (mandatory) attributes for GeographicArea and its four subtype units shown in Figure 3.5.1, as described in the data dictionary tables in Part 5.

- *unitId*: permanent identifier assigned to the unit
- *instanceName*: "Official" feature name. If available, the instanceName is the name of the geographic area feature from the Geographic Names Information System (GNIS)
- *instanceCode*: Specific code that identifies the geographic area
- *codingSystemReference*: citation, reference, or documentation identifying the instance code type, for example USPS ZIP+4 code
- *geometry*: either GM_Surface (e.g. polygon). Listed as optional in the data dictionary because the Boundary class can be used instead to represent the boundary segments of the unit

Attributes specific to whether the unit is a GovernmentalUnit, AdministrativeUnit, StatisticalUnit or OtherUnit:

- *[one of the four types]unitType*: depending on which unit the feature is, this attribute uses values from the appropriate codelist. Examples for GovernmentalUnit include state, county, tribalNation, city, village, etc.
- *typeDefinition*: if a value for *unitType* is used that is not included on the codelist for *unitType*, this attribute is required in order to provide a definition of the unit type.

Refer to section 8.11 of Part 5: Governmental Units for the complete codelists for unitType and their definitions.

### 3.5.2 Maintenance relationships between units (optional)

A maintenance relationship exists when there is a relative dependency between two or more geographic areas for maintaining common boundary or area information. For instance, if there are two school districts adjacent to each other, what happens when you change the boundary of one school district? What happens to the other school district if it shares the boundary that is changed?

Determination types specify how changes to a shared boundary by one geographic area (i.e. unit) affect that boundary from the perspective of the other geographic area. Two determination types are explained here. For additional types, see section 6.3 of Part 5 which also includes graphical representations of the relationships, or see the DeterminationType codelist in Table 12 of Part 5.

- Co-determined: Changes to a boundary segment by either geographic area that shares the boundary segment imposes the same changes to both geographic areas. This results a *ChangeType* of “expansion” for one geographic area and a spatial change type of “contraction” for the other.

- Independent of: Changes to a boundary segment by one geographic area creates a new, unshared boundary segment in the boundary of that geographic area, and impose no changes to the boundary of another geographic area that shared the boundary segment by coincidence.

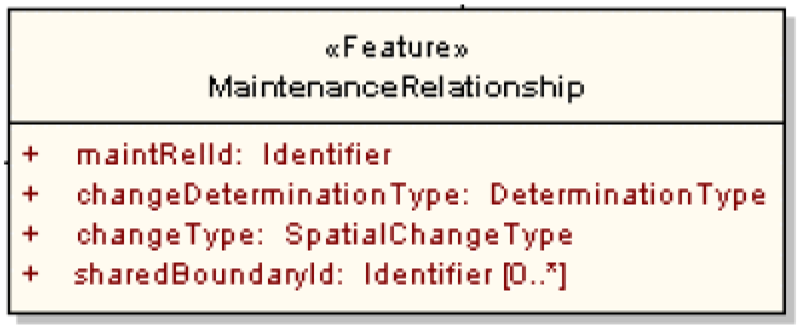

«Feature»
MaintenanceRelationship
+ maintRelId: Identifier
+ changeDeterminationType: DeterminationType
+ changeType: SpatialChangeType
+ sharedBoundaryId: Identifier [0..*]

**Figure 3.5.2**
**MaintenanceRelationship class and its associated attributes.**

An example of an implementation of a maintenance relationship in a relational database is shown in Table A.3 of Annex A in Part 5: Governmental Units.

### 3.5.3 Spatial relationships between units (optional)

Spatial relationships describe the relative spatial location of two geographic area features and their boundaries. For instance, counties are contained by states. Census blocks are contained by census block groups which are in turn contained by census tracts.

Some types of administrative units have the potential to overlap each other. For instance, parks may overlap with precincts.

Units that share a boundary use the spatial relationship term “touch”.

For additional relationship types, see section 6.4 of Part 5, which also includes graphical representations of the relationships. Table 16 of Part 5 is the full SpatialRelationshipList enumeration with definitions.

An example of an implementation of spatial relationships in a relational database is shown in Table A.2 of Annex A in Part 5: Governmental Units.

## 3.6 Framework Data Content Standard Part 6: Hydrography

### 3.6.1 Introduction

Hydrographic data is information about surface water features such as lakes, ponds, streams, rivers, springs, marshes and even glaciers. It can also include man-made features such as canals, ditches, pipelines, and reservoirs, and water-related features such as dams, bridges, wells and diversions. This data is often collected in various forms, such as remote sensing and field work, and it can also be derived from elevation and other surface features.

On maps or in digital databases, hydrographic data can be represented as lines (e.g. streams, canals, pipelines), areas (rivers, lakes, marshes, reservoirs), and points (springs, wells). Scale will affect how hydrographic features are represented. For instance at small scales covering large areas, rivers are usually represented as lines; at large scales rivers and streams may be represented in more detail as areas with width. Multiple representation of a feature may also occur due to changes over time. For instance changes due to irrigation or construction may alter the path of streams. The boundaries of lakes or reservoirs may change as water levels rise or fall due to operation of dams, or periods of drought.

Complex or compound features are another characteristic of some types of hydrographic data. For instance, a stream is usually composed of many segments along its length, where segments usually occur when another stream or tributary flows into it. Reaches and watercourses are types of hydrographic features, usually composed of more than one stream segment, grouped together for a specific purpose such as for sampling or navigation.

Hydrographic data is often required to describe the interconnectedness of a surface water system. Some applications may also require that hydrographic data have flow direction associated with it. For example, if attributes specify that one feature flows from another feature and to another feature, then operations such as tracing movement upstream or downstream can be performed. These types of applications usually require the use of centerlines to maintain connectivity and flow direction through features with area, such as wide rivers, lakes and reservoirs.

Because of the linear nature of some types of hydrographic features (e.g. streams, rivers, canals, pipelines), linear referencing is another characteristic frequently associated with this type of data. A Linear Referencing System (LRS) is a reference system in which the locations of features are determined by measures along a linear element. For instance, a portion of a river may be sampled for water quality or aquatic species/habitat. The sampling location can be identified by measures along the river instead of as a separate linear feature or by creating new segments along the river. The system is designed so that if the sampling location is changed only the measurements need to be updated, instead of the coordinates of the beginning and end points. Linear referencing systems are used for efficiency in terms of data updates and data size.

All of these characteristics of hydrographic data are taken into account in the Framework Standard Part 6: Hydrography, though they are included as optional elements. This part identifies and defines terminology, data components and their relationships required for describing hydrographic features, along with the metadata needed to enable collaborative development, use, and exchange of hydrographic data. The Framework Standard does not specify a particular structure for the data, but rather provides a standardized way of describing the content. It is up to the data producers to decide which characteristics to include, including complex or compound features, flow direction, and linear referencing.

The *FGDC online training materials webpage* has materials for download (PowerPoint slides) specific to Part 6: Hydrography. These materials include other graphic examples of hydrography, as well as descriptions and links to several websites that describe or serve hydrography data.

### 3.6.2 Relationship to the National Hydrography Dataset (NHD) and other hydrographic databases

This part of the Framework Standard, and the included UML model, is a result of contributions from a variety of information and systems models. These include: the National Hydrography Dataset (NHD), the Pacific Northwest Framework (PNW), the ArcHydro data model, and the Geographic Names Information System (GNIS).

As stated in the Purpose (section 1.2) of the Hydrography Part of the Framework Standard, the intent of this Part is to set a common baseline of information content for exchange within the hydrographic community that will enhance data sharing and applications development when used with standards-based Web services or file transfer. The developers of this Part acknowledge that multiple representations of hydrographic features exist within the broader community and that policies have been or will be established for describing, maintaining, and exchanging the various representations of features within specific application communities, such as the NHD.

The Framework Standard does not intend to replace existing application structures such as the NHD, but seeks to further accommodate exchange of data by implementing efforts of ISO and OGC towards interoperability. This Part supports the mapping and conversion of native data in any format into a common representation for exchange and integration. Because of the important role of the NHD in the National Map (another part of the National Spatial Data Infrastructure, complementary with Framework efforts), subsequent subchapters provide specific examples of mapping data components of the NHD to the Hydrography Part of the Standard.

### 3.6.3 Requirements

The Requirements (section 6) of the Framework Standard, Part 6: Hydrography includes an overall UML class diagram of the Hydrography application schema and data dictionary tables that further describe the UML objects. The Requirements section also provides codelists and enumerations that list standard values for attributes specified in the data dictionary tables. The following subchapters of this document describe each of the classes in the UML diagram for hydrography and selected descriptions from the data dictionary tables, along with examples. Examples from codelists and enumerations are also included so that the hydrography requirements can be understood without having to flip back and forth between different sections as they are organized in the Hydrography part of the Framework Standard.

Figure 3.6.1, Hydrographic features and events, shows only selected classes from the complete UML class diagram for hydrography (Figure 1 from Part 6: Hydrography). The selected classes are shown for simplification. The rest of the classes are discussed in later subchapters of this chapter.

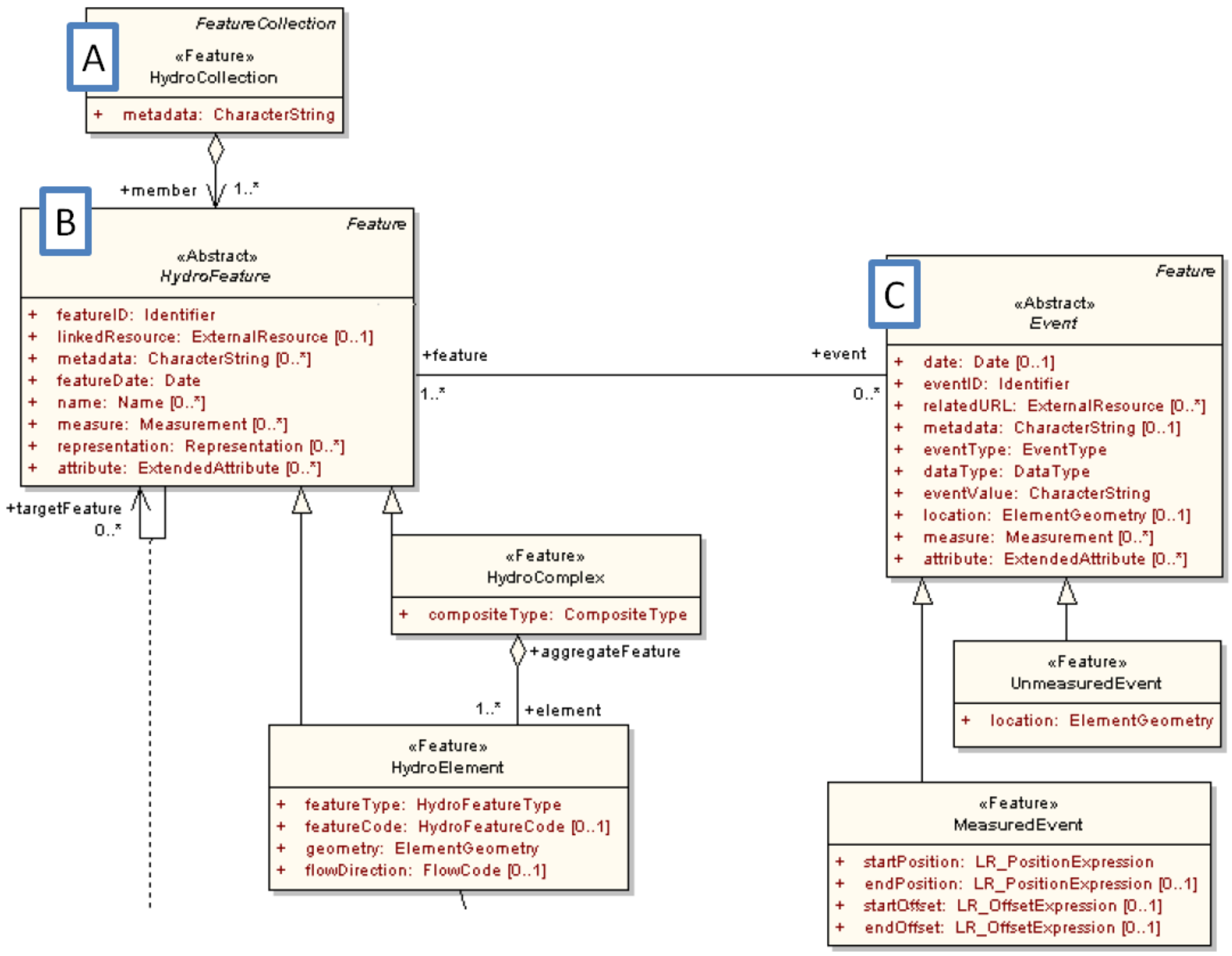

A
FeatureCollection
«Feature»
HydroCollection
+ metadata: CharacterString
+member
1..*
B
Feature
«Abstract»
HydroFeature
+ featureID: Identifier
+ linkedResource: ExternalResource [0..1]
+ metadata: CharacterString [0..*]
+ featureDate: Date
+ name: Name [0..*]
+ measure: Measurement [0..*]
+ representation: Representation [0..*]
+ attribute: ExtendedAttribute [0..*]
+feature
1..*
+event
0..*
C
Feature
«Abstract»
Event
+ date: Date [0..1]
+ eventID: Identifier
+ relatedURL: ExternalResource [0..*]
+ metadata: CharacterString [0..1]
+ eventType: EventType
+ dataType: DataType
+ eventValue: CharacterString
+ location: ElementGeometry [0..1]
+ measure: Measurement [0..*]
+ attribute: ExtendedAttribute [0..*]
+targetFeature
0..*
«Feature»
HydroComplex
+ compositeType: CompositeType
+aggregateFeature
1..*
+element
«Feature»
HydroElement
+ featureType: HydroFeatureType
+ featureCode: HydroFeatureCode [0..1]
+ geometry: ElementGeometry
+ flowDirection: FlowCode [0..1]
«Feature»
UnmeasuredEvent
+ location: ElementGeometry
«Feature»
MeasuredEvent
+ startPosition: LR_PositionExpression
+ endPosition: LR_PositionExpression [0..1]
+ startOffset: LR_OffsetExpression [0..1]
+ endOffset: LR_OffsetExpression [0..1]

**Figure 3.6.1**
**Hydrographic features and events (detail from Figure 1 in Part 6: Hydrography)**

Figure 3.6.1. shows three primary classes, <<Feature>> HydroCollection (A), <<Abstract>> Hydrofeature (B), and <<Abstract>> Event (C).

<u>HydroCollection class</u>
<<Feature>> HydroCollection (A) is an instance of FeatureCollection, from the overall Framework application schema specified in the Part 0: Base. It represents a collection of hydrographic features and/or events requested for data transfer.

<u>Abstract classes</u>
Hydrofeature (B) and Event (C) are both abstract classes, which means they are generalizations and there are no direct instances of either of these classes. They are used to identify attributes in common that are inherited by their subtypes, which are actual instances of Features (HydroElement and HydroComplex) or Events (MeasuredEvent and UnmeasuredEvent).

HydroElement and HydroComplex classes
For example, flowlines in the NHD can be mapped to a HydroElement. Figure 3.6.2 shows an example of NHD flowlines, each segment with its own ID number (1 through 9). Each individual stream segment corresponds to a HydroElement.

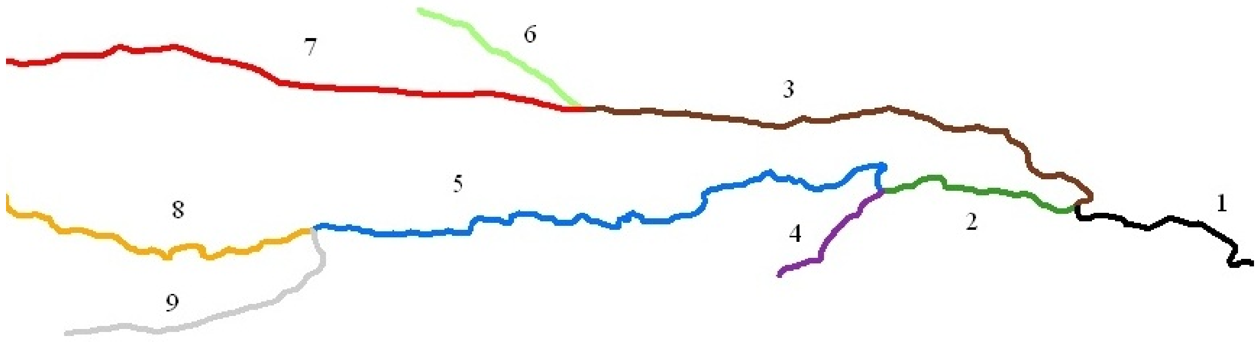


**Figure 3.6.2**
**NHD flowlines which correspond to a total of 9 individual HydroElements**

Figure 3.6.3 shows the same example, now labeled as NHD reaches, where some segments have been grouped together to form a reach (e.g. South Fork Mill Creek is made up of three segments (2,5, and 8 from Figure 3.6.2). Each reach corresponds to a HydroComplex, even the unnamed single segments.

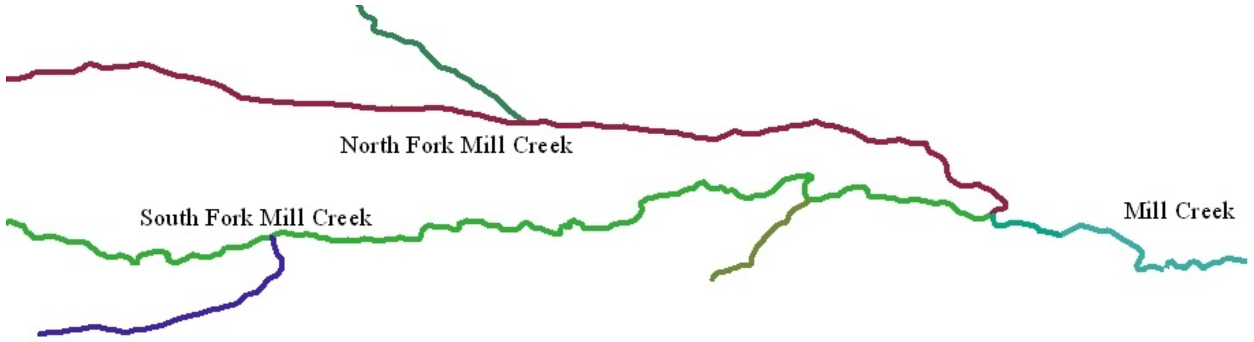


**Figure 3.6.3**
**NHD reaches which correspond to a total of 3 named HydroComplexes and 3 unnamed HydroElements.**

HydroComplexes and HydroElements share the same attributes, including several mandatory attributes: *featureID*, *featureDate*, *featureType* and *geometry*.

In addition, HydroComplex as another mandatory attribute, *compositeType* which takes it value from the CompsiteType codelist. Two values, NHDReach and watercourse, are included in the codelist but additional codes can be added as required.

Other attributes for HydoElements and HydroComplexes are optional, as indicated by the [0..1] notation. A list attributes is provided in Table 3.6.1, along with examples and the matching NHD attribute.

**Table 3.6.1**
**Attributes for HydroFeature and HydroElement with equivalent NHD attribute (if it exists) and example values. Attributes listed in bold in the first column are mandatory, and attributes listed in bold in the second column are mandatory if applicable.**

| Name | Inherited attributes, codelist or enumeration | Maximum occurrence | Matching NHD attribute | Example value(s) |
|---|---|---|---|---|
| HydroFeature | | | | |
| **featureID** | ***id*** (+other optional attributes) | 1 | ComID | 113175895 |
| **featureDate** | | 1 | Fdate | 12/4/2003 |
| linkedResource | | 0..* | | |
| Metadata | | 0..* | | |
| Name | | 0..* | GNIS_Name | Muddy Creek |
| measure [0..*] | ***reportingOrganization*** | 1 | | USGS/NHD |
| | ***units*** (UnitsType codelist) | 1 | | LengthKM |
| | ***value*** | 1 | | 4.147 |
| | *accuracy* | 0..1 | | |
| Representation | ***id*** (+other optional attributes) | 1 | ReachCode | 14050004001836 |
| attribute [0..*][1] | ***authority*** | 1 | | NHD/USGS |
| | ***name*** | 1 | | Resolution |
| | ***type*** | 1 | | characterString |
| | ***value*** | 1 | | high |
| HydroElement | | | | |
| **featureType** | HydroFeatureType codelist | 1 | Ftype | streamRiver |
| **Geometry** | ***line*** | 1 | Shape | GM_Line |
| featureCode | | 0..1 | Fcode | Stream/River: Hydrographic Category = Perennial |
| Flowdirection | FlowCode enumeration | 1 | FlowDir | flowsWith[2] |

[1] attribute is an ExtendedAttribute type, for including attributes that are not part of the Framework data model
[2] the corresponding NHD value is "WithDigitized"

All parts of the Framework Standard, not just hydrography, emphasize permanent features with unique permanent IDs to support the community's uses of data. See subchapter 2.3.1.5 of this document for more information about permanent identifiers and their importance.

### 3.6.4 Optional classes for hydrography

This subchapter includes a discussion of these optional classes: Events, FeatureRelationships, and ComputedNetworkValues, and how they are related to some of the implementation design requirements mentioned in Appendix B of the Hydrography part of the Framework Standard. In addition, discussion is given to how this part of the Framework Standard may be implemented with regard to multiple representations of features (e.g. the same feature being represented at different scales or at different periods of time).

#### *3.6.4.1 Events*

Events are data that are linked to HydroElements or HydroComplexes, and have a location that falls on or across one or more features. For instance, rapids are an event that can occur along a stream, encompassing parts of multiple stream segments (HydroElements). The difference between a MeasuredEvent and an UnMeasuredEvent is determined by how the location of the event is referenced.

A MeasuredEvent is located by linear referencing. A linear HydroElement or HydroComplex is given an absolute or relative measure. For instance, the NHD uses a relative measure, where each reach is assigned a measure from 0 to 100. The start of the measure is 0 at its confluence with another reach, and ends at 100 its source.

If the MeasuredEvent type is linear (has a length, such as a stretch of rapids), it will have a startPosition and an EndPosition which corresponds to whatever type of measure is used on the reference features. In our NHD example, rapids might have a startPosition at 25 and an endPosition at 50 along the stream (Figure 3.6.4). If the event type is not linear, such as a bridge, it will have only a startPosition, such as the bridge at 75 (approximately three quarters of the way along the reach). This is only one example of implementing the measurement of events. Absolute measures of a HydroElement or HydroComplex might begin at 0 km and end at 4 km, where a bridge might occur at 3 km.

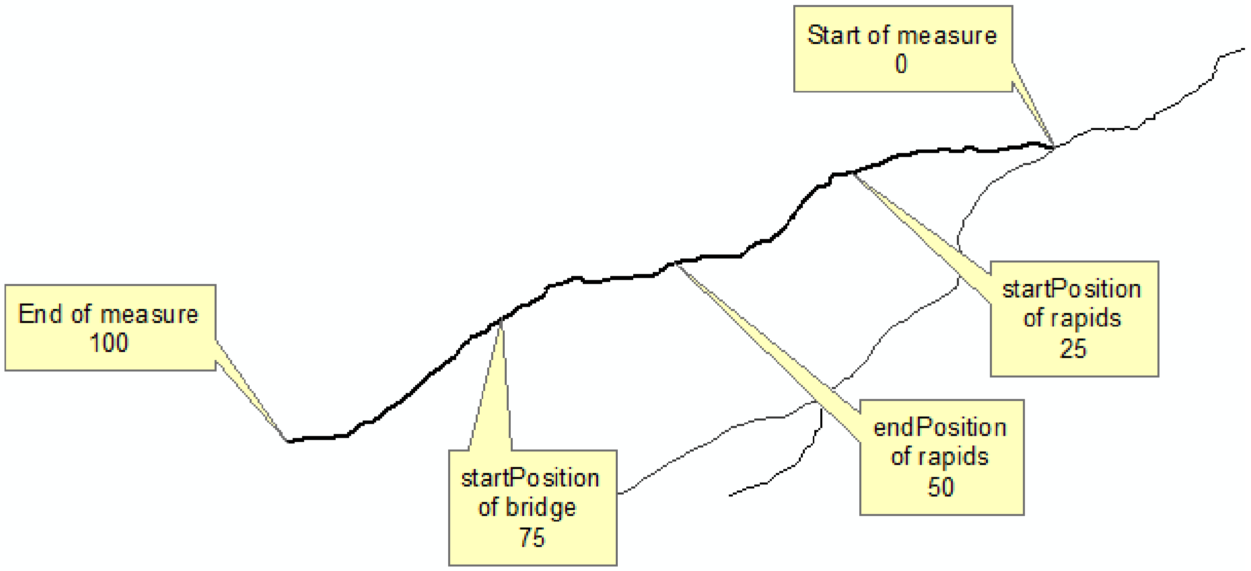


**Figure 3.6.4**
**The linear referencing of a point event (bridge) and a linear event (rapids) along a reach, using relative measures.**

An UnmeasuredEvent will have its own geometry (x,y coordinates), stored separate from the geometry of the HydroElement or HydroComplex that it references. For instance, NHD geodatabases usually contain point events (such as dams or gaging stations), line events and area events.

Both MeasuredEvents and UnmeasuredEvents are linked to HydroFeatures by a value that uniquely identifies each feature or composite feature. For instance, the point events in the NHD all have a ReachCode value that matches the ReachCode in the NHD flowlines.

The mandatory attributes required for an Event are:

- *eventID*: a unique identifier for the event (this corresponds to the attribute ComID in the NHD)
- *eventType*: examples from the EventType codelist are “bay/inlet”, “bridge”, “rapids”, “waterfall”
- *eventValue*: value associated with the event
- *dataType*: data type for the information stored in the eventValue (e.g. characterString, integer)
- *location*: geometry required for an UnmeasuredEvent
- *startPosition*: a measure required for a MeasuredEvent

### *3.6.4.2 FeatureRelationships*

This part of the model describes relationships between features in order to describe connectivity and flow of streams. In Figure 3.6.5, the arrow from HydroFeature that points back to itself means that Hydrofeatures can participate in relationships (e.g flow to) other HydroFeatures.

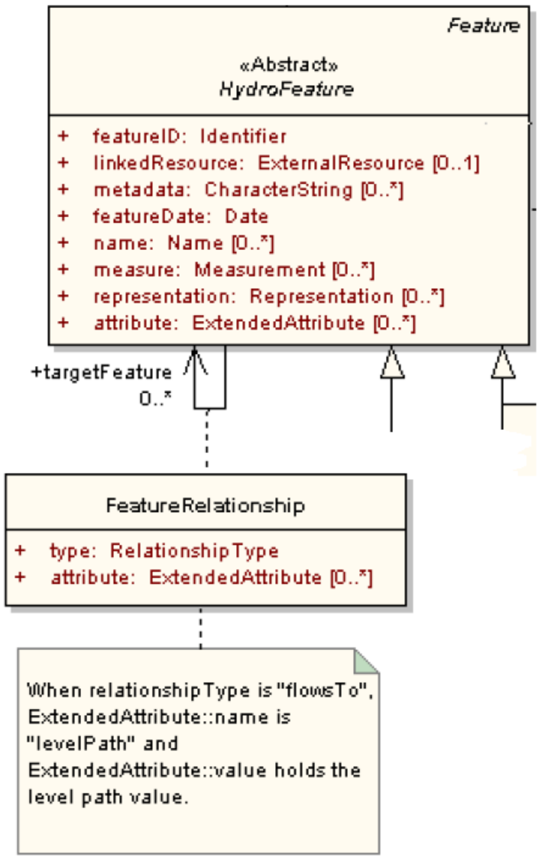

Feature
«Abstract»
HydroFeature
+ featureID: Identifier
+ linkedResource: ExternalResource [0..1]
+ metadata: CharacterString [0..*]
+ featureDate: Date
+ name: Name [0..*]
+ measure: Measurement [0..*]
+ representation: Representation [0..*]
+ attribute: ExtendedAttribute [0..*]
+targetFeature
0..*
FeatureRelationship
+ type: RelationshipType
+ attribute: ExtendedAttribute [0..*]
When relationshipType is "flowsTo", ExtendedAttribute::name is "levelPath" and ExtendedAttribute::value holds the level path value.

**Figure 3.6.5**
**FeatureRelationship between HydroFeatures (detail from Figure 1 in Part 6: Hydrography)**

Relationships are optional, as indicated by the [0..1] notation next to the arrow.

If a relationship does exist, it must have a mandatory attribute *type*, which uses codes from the RelationshipType codelist. The codelist includes (but is not limited to) the following:

- flowsTo: a downstream relationship between the first, or “from”  feature to the second or “to” feature.
- Overpass: relationship between features where the first feature overpasses the second feature such that the water flows do not combine. For example, a pipelines crossing over a stream

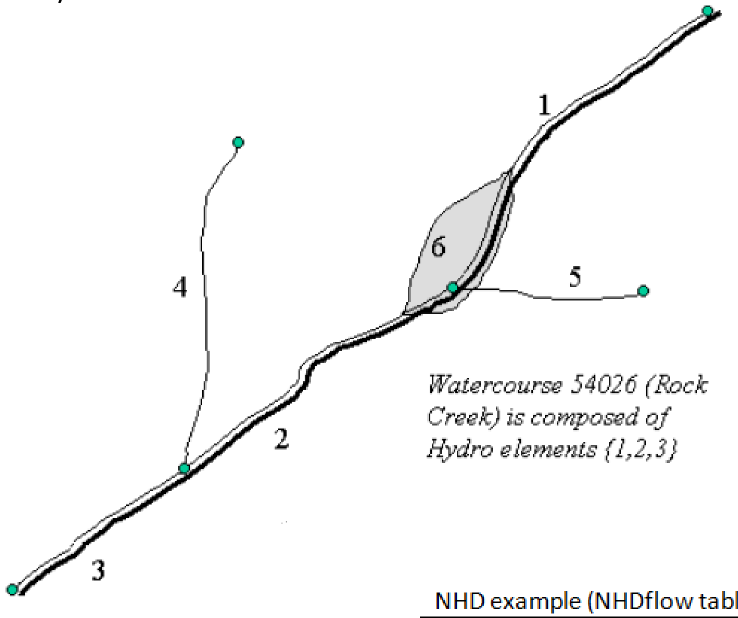
In addition to the two values described above in the RelationshipType codelist, Appendix B of this part of the Framework Standard mentions a third possible relationship that could be implemented: “flowsThrough.” This could be used to describe the relation between artificially derived flow paths and the lakes, ponds or reservoirs that they flow through in order to create a connection between inflowing and outflowing streams. These artificial paths maintain connectivity and flow direction for purposes of navigation or network analysis.

Relationships can be implemented several ways. The NHD originally stored them in a table with IDs of “From” and “To” features. In the NHD geodatabase model (NHDinGEO), the relationships are stored in the ArcGIS geometric network data model, rather than in a relational table.

Figure 3.6.6 shows an example with six HydroElements (modified from the example provided as Figure 3 in Part 6: Hydrography), and two different examples of how "flowsTo" relationship might be implemented. Feature 1 (stream) flows to Feature 6 (lake/pond), which corresponds to the first row in the Framework example, and the second row in the NHD example. The first row in the NHD example shows how NHD indicates that Feature 1 is a headwater, because *the FromComID* value is 0, indicating it does not flow from any other feature.

The NHD table in Figure 3.6.6 does not have a corresponding attribute to the *relation* attribute in the Framework example; this is one example where NHD does not match to Framework (the "flowsTo" relationship is implied rather than explicitly stated).

The NHD example does, however, provide a *DeltaLevel* attribute which can be matched to the *levelPath* attribute described in the model shown in Figure 3.6.5. If DeltaLevel is 0, it means both features in the relationship are at the same stream level. Stream level is defined in this part of the Framework Standard as "level within a stream classification system based on the position of the stream within a drainage Etwork." It is identified by a numeric code such that streams that terminate in sea/ocean features are assigned to the lowest level (the Mississippi river is Level 1) and tributaries are incremented based on the level into which they terminate at their headwaters.


1
6
5
4
2
3
Watercourse 54026 (Rock Creek) is composed of Hydro elements {1,2,3}

Framework example

| elementA | elementB | relation |
|---|---|---|
| 1 | 6 | flowsTo |
| 5 | 6 | flowsTo |
| 6 | 2 | flowsTo |
| 2 | 3 | flowsTo |
| 4 | 3 | flowsTo |

NHD example (NHDflow table)

| FromComID | ToComID | DeltaLevel |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 6 | 0 |
| 5 | 6 | 0 |
| 6 | 2 | 1 |
| 2 | 3 | 1 |
| 4 | 3 | 1 |

**Figure 3.6.6**
**An example of a FeatureRelationship "flowsTo", with both a Framework example and an NHD example.**

In Figure 3.6.6, linear features 1 and 2 are shown flowing through a lake/pond (feature 6). In the NHD, this is modeled differently than the example. A linear feature called “artificial path” would exist inside the lake/pond. Thus, feature 1 would terminate at the edge of the lake/pond, with a “flowsTo” relationship to an artificial path feature, rather than into the lake/pond feature. Then the artificial path feature would have a “flowsTo” relationship to feature 2, which would have its starting point at the edge of the lake/pond. Both methods of implementation, the example shown above in Figure 3.6.6 and the artificial path feature method described for NHD, are valid ways of implementing this part of the Framework Standard.

#### *3.6.4.3 ComputedNetworkValues*

The HydroElement class has an optional attribute called flowDirection. There is a corresponding enumeration list called FlowCode (recall that an enumeration is a fixed list of values that cannot be extended with additional values like a codelist). These are the following acceptable FlowCodes and their descriptions:

- notApplicable (0): flow inference not applicable to the feature, e.g. shoreline
- flowsWith (1): flow of water is the same direction as the coordinate order
- flowsOpposite (2): flow of water is the opposite direction of coordinate order
- unknown (3): flow direction of water unknown
- bidirectional (4): water may flow either direction along feature, e.g. certain types of pipelines)

In Figure 3.6.7, if FlowCode = 1 (flowsWith) for a HydroElement then there may be optional [0..1] instances of ComputedNetworkValues. The NHD includes a number of computed values for hydrographic features (HydroElements) with basic geometries.

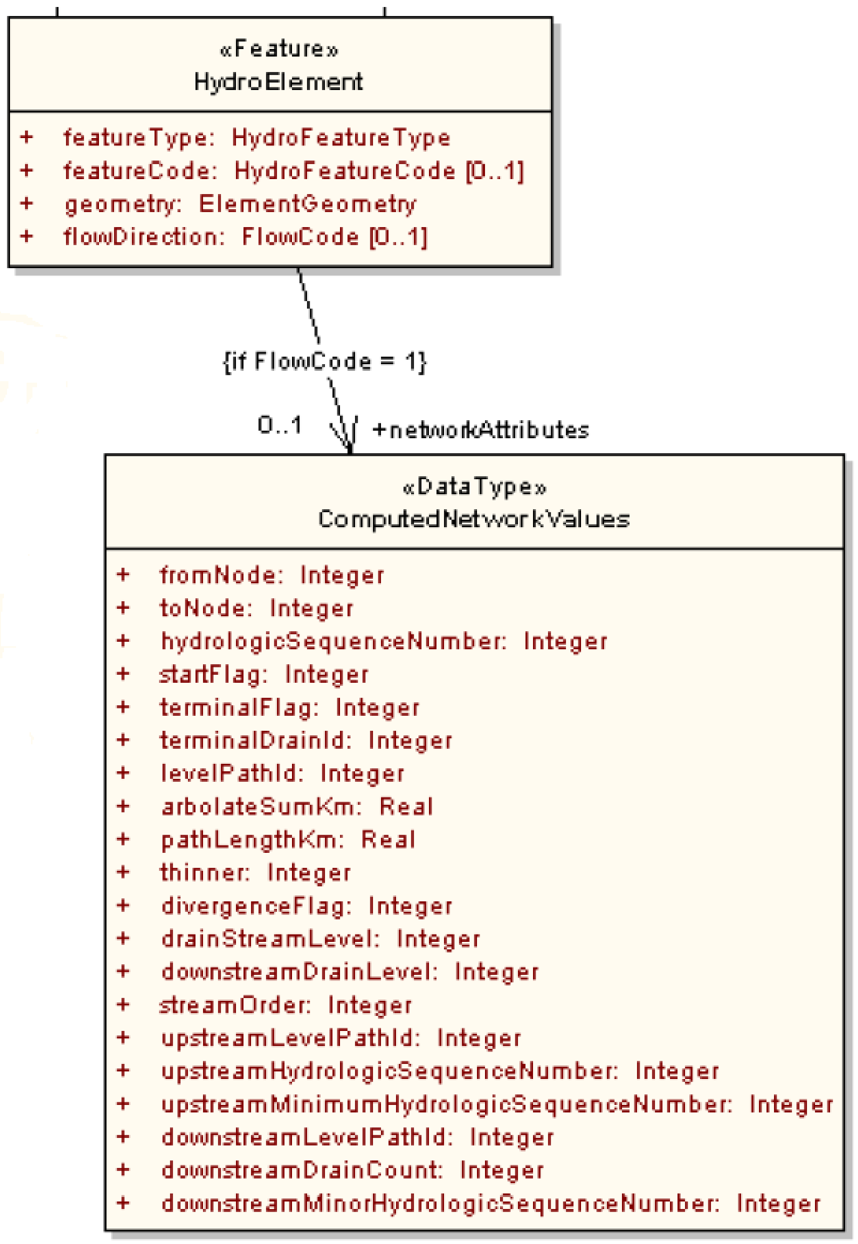

«Feature»
HydroElement
+ featureType: HydroFeatureType
+ featureCode: HydroFeatureCode [0..1]
+ geometry: ElementGeometry
+ flowDirection: FlowCode [0..1]
{if FlowCode = 1}
0..1
+networkAttributes
«DataType»
ComputedNetworkValues
+ fromNode: Integer
+ toNode: Integer
+ hydrologicSequenceNumber: Integer
+ startFlag: Integer
+ terminalFlag: Integer
+ terminalDrainId: Integer
+ levelPathId: Integer
+ arbolateSumKm: Real
+ pathLengthKm: Real
+ thinner: Integer
+ divergenceFlag: Integer
+ drainStreamLevel: Integer
+ downstreamDrainLevel: Integer
+ streamOrder: Integer
+ upstreamLevelPathId: Integer
+ upstreamHydrologicSequenceNumber: Integer
+ upstreamMinimumHydrologicSequenceNumber: Integer
+ downstreamLevelPathId: Integer
+ downstreamDrainCount: Integer
+ downstreamMinorHydrologicSequenceNumber: Integer

**Figure 3.6.7**
**ComputedNetworkValues (detail from Figure 1 in Part 6: Hydrography)**

These properties are only calculated for features that participate in the network for which flow direction is known or inferred, and are typically found in the NHD table called NHDFlowlineVAA. Though there is some variation between the attribute names in the NHD table and the attribute names listed for ComputedNetworkValues shown in Figure 3.6.7, the description between the NHD attributes and data dictionary for ComputedNetworkValues (Table 12 in Part 6: Hydrography) are close enough to be matched. Many of these calculated values are used for navigation and various types of network analysis, such as upstream or downstream tracing.

### 3.6.4.4 *Multiple representations of features*

Part 6: Hydrography supports the management of multiple representations of features. A feature instance may have multiple representations reflecting different geometries and attributes due to changes over time (updates or corrections), changes in scale, or differing generalization criteria applied in support of user needs.

Some feature types, such as reaches and watercourses, do not have direct spatial representation, but instead derive their geographic extent through association to other feature types that do have spatial representation (individual segments of streams). Therefore, a unique representation identifier is required to identify the specific instance of a feature – including its specific attributes such as a date, and optional attributes such as name and even possibly an alternative geometry.

This representation identifier provides a unique code to identify the state of the described feature. For example, a change in attribution or geometry preserves the same *permanent identifier* for purposes of linking to references or tracking changes, but would be tagged with a new *representation identifier* to indicate a change in the representation.

An example of this can be seen in the NHD table called NHDReachCrossReference. This table contains two attributes that are representation identifiers: *OldReachCode* and *NewReachCode*, along with additional attributes that describe the date of change and the specific process that results in the change, such as the deletion of old reachcodes, the addition of new reachcodes, or the translation of an old reach to part of a new reach.

Another example of this can be seen in the NHD, where there are two related NHD tables, NHDFeatureToMetadata and NHDMetadata. The first table contains a list of unique identifiers for NHD flowlines (which correspond to Framework HydroElements) wherever there is specific feature-level metadata related to the flowlines; the actual metadata is contained in the second table. For instance, some features are tagged with metadata that indicates their geometry was corrected after QAQC using USGS 24k DRGs (topographic backgrounds) and NRCS NAIP (orthorectified aerial photos). This implementation fits the Framework Standard since a new *representation identifier* (the NHD attribute DuuID) was created in order to identify these changes.

As with the *featureID* and *eventID*, the *representationID* also uses the Identifier data type, inherited from the overall Framework application schema in the Part 0: Base. The unique identifier can be accompanied by optional *idAuthority* and *description* attribute.

## 3.7 Framework Data Content Standard Part 7: Transportation Base

### 3.7.1 Introduction

Transportation can be defined as the movement of people and goods from one place to another. For example a state highway map, railroad timetable, airline routes are all examples of transportation data. Many real-world transportation features are linear in nature. For clarification in this part of the Framework Standard, linear refers to objects with length, usually one-dimensional, but not limited to straight lines. For instance, roads are often represented as linear features that do not have a width (the width can be described as an attribute). Roads can be either straight or curved lines. This part of the Framework Standard provides means for describing transportation features that are linear in nature and that have connectivity with other linear features, for the purposes of describing routes or paths. It also allows representation of non-linear features, such as points, or features with area.

The Transportation Base part integrates five modes of transportation systems: air (Part 7a), rail (Part 7b), road (Part 7c), transit (part 7d), and water (Part 7e). Part 7a, for air, has not yet been endorsed by the FGDC at the time of publication of this guidance document, and is not included in this document. The other four transportation parts are included in subchapters 3.7.5 through 3.7.9 of this document.

There are several applications of transportation data addressed by the Transportation Base part of the Framework Standard.

1) The Framework Transportation model is accommodates data modeled as a transport network. Transport network analysis to determine the flow of vehicles (or people) through it, including path finding/routing applications. It may combine different modes of transport, such as air travel, railway, and roads to model multi-modal journeys. For instance, a railway station that includes connections to roads can be modeled several different ways. It could be modeled as both a rail feature and a road feature. Alternatively, it could be modeled as a transportation feature. Modeling a railway station as the more generalized transportation feature allows it to be a connection point to other modes of transportation for the purposes of transport network analysis.

2) The Framework Transportation model accommodates data encoded with geometry (for use with Geographic Information Systems) and also data encoded without geometry. The reason why data encoded without geometry is important because some path finding or routing applications can produce faster results if they do not have to process geometry; results can be determined based on the topology (connectivity of features to each other, or other relationships) and do not require specific geometries.

3) The Framework Transportation model accommodates assets associated with the transportation system that are typically used for navigation, safety, and measurement. This means that the application schema (e.g. data model) for transportation can include features that are not actual transportation features (e.g. roads, railways, transit lines) but are important to the proper and safe functioning or navigation of the transportation system. Examples include signs, signals, traffic measure devices, switches, platforms, communication towers, and so forth.

4) The Framework Transportation model accommodates a Linear Referencing System (LRS). The Transportation Event model defines events as attributes or entities that can be linearly located along either a transportation feature. For instance, a road may have a speed limit of 55 mph for its first two miles and a speed limit of 65 mph for the remainder of its length. A linear referencing system (using mile posts or some other measure) allows the extent of both speed limits to attributed to one road feature and may also allow these attributes to be visualized in a GIS.

5) The Framework Transportation model accommodates the need for defining equivalencies between transportation features (particularly roads) that are represented differently in disparate databases. For example, a database from one agency may have different positional accuracies, different linear reference methods (LRMs), or different schemes for partitioning (segmenting) the transportation network than a database from another agency. End users may have a variety of compelling business needs to distinguish each representation but must also know that each is a representation of the same transportation system. "Equivalence relationship" is a correlation used to indicate that a transportation feature from one source is equivalent to (that is to say, has the same physical location as) one or more transportation features from another source. More information about equivalencies is included in subchapter 3.7.2.3 and also in the informative annex provided in Part 7: Transportation Base document (Annex A).

Users and producers of transportation data will benefit from using the Transportation parts of the Framework Standard in several ways. In addition to the general benefits of the Framework Data Content Standard discussed in subchapters 1.4 and 1.5 of this guidance document, the Transportation Base part identifies additional benefits specific to transportation, including improved integration of safety, emergency response and enforcement data. Being able to link accident information to transportation data allows specialists to analyze areas that are prone to a higher frequency of accidents, and the ability to integrate transportation data with other data such as traffic measures can identify potential reasons for increased accidents and ways to prevent accidents. Information related to enforcement of speed limits, turning rules or other transportation rules can also help increase transportation safety and assist in analysis of traffic routing, both for day-to-day and also in emergency or disaster situations. The Framework Transportation data model provides a structure and standardized terms for incorporating safety, navigation and enforcement data with transportation features.

### 3.7.2 Transportation Data Model

The Transportation Base part includes several UML class diagrams and accompanying data dictionaries for the Transportation Base model showing all Transportation classes and their relationships. For simplicity, this subchapter focuses on subparts of this model, particularly the Transportation Feature model and the Transportation Event model. Please refer to Figure 1 in the Transportation Base part document to see the entire model. The following subchapters provide a description of major classes of the UML model, mandatory elements of the data dictionaries, definitions of terms, and examples. This guidance document provides the same information as the Part 7: Transportation Base document, but condenses it and pulls information that is found in many different sections and annexes of the Transportation Base document into a summary format. The following subchapters provide references to the appropriate parts of the Transportation Base document for more details.

### 3.7.2.1 Transportation Feature Model

At the most general level, the TranFeature class is used to model transportation features (see Figure 3.7.1). Three subtypes of this class are used for handling linear features with connectivity: TranPath, TranSeg, and TranPoint. These three feature subclasses have analogues in the rail, road, transit, and waterway modes of transportation.

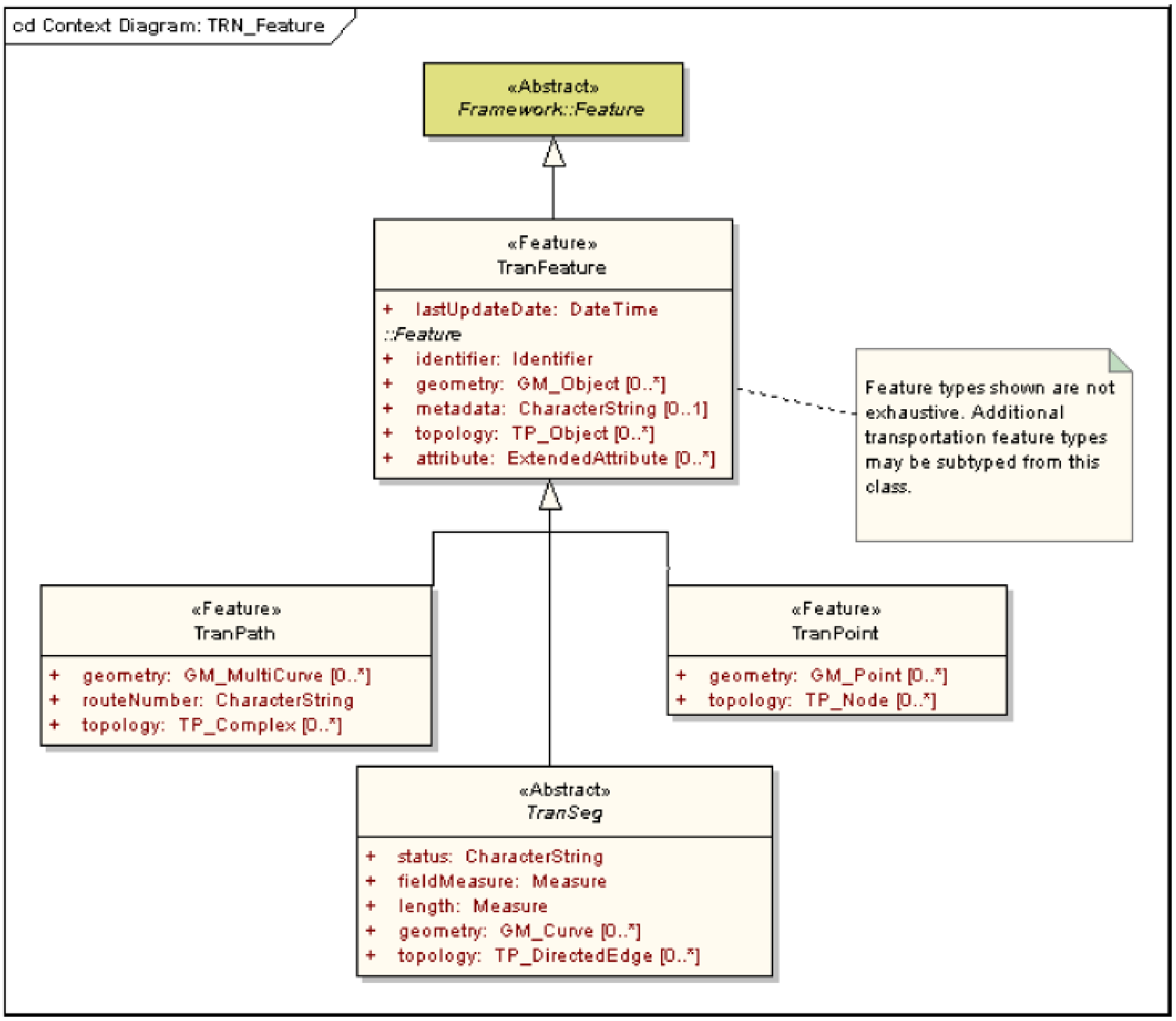

cd Context Diagram: TRN_Feature
«Abstract»
Framework::Feature
«Feature»
TranFeature
+ lastUpdateDate: DateTime
::Feature
+ identifier: Identifier
+ geometry: GM_Object [0..*]
+ metadata: CharacterString [0..1]
+ topology: TP_Object [0..*]
+ attribute: ExtendedAttribute [0..*]
Feature types shown are not exhaustive. Additional transportation feature types may be subtyped from this class.
«Feature»
TranPath
+ geometry: GM_MultiCurve [0..*]
+ routeNumber: CharacterString
+ topology: TP_Complex [0..*]
«Feature»
TranPoint
+ geometry: GM_Point [0..*]
+ topology: TP_Node [0..*]
«Abstract»
TranSeg
+ status: CharacterString
+ fieldMeasure: Measure
+ length: Measure
+ geometry: GM_Curve [0..*]
+ topology: TP_DirectedEdge [0..*]

**Figure 3.7.1**
**Transportation feature type hierarchy (reproduced from Figure 2 in Part 7: Transportation Base document).**

All other non-linear transportation related features can be modeled as TranFeatures, with the option to create user-defined TranFeature subtypes. In addition, transportation-related features can also be modeled as FeatureEvents, which allows important network features such as traffic lights, railroad switches, etc. to be included as part the linear network, even though they are not linear features. FeatureEvents are discussed in subchapter 3.7.2.2.

Regardless of whether transportation features are linear or not, they are all required to have an *identifier* attribute, a means of permanent identification, and a *lastUpdateDate* attribute indicating when the TranFeature was last edited. All other attributes are optional. The optional *Geometry* attribute uses the GM_Object class defined in ISO 19107 which can include point, line, polygon geometry and other variations. The optional *Topology* attribute is used to describe relationships between features (such as connectivity) and is used for several purposes, such as to speed up computationally intense geometric calculations between objects, or to relate objects to each other independent of their geometry. Examples of topological relationships: object A is connected to object B; Object A is adjacent to Object B; Object A is contained by Object B. For a more detailed discussion of topology, please refer to ISO 19107: Geographic Information – Spatial Schema.

The TranPath, TranSeg and TranPoint subtypes and their relationships are used together to define all possible movements through the transportation system. Examples of applications using these subtypes would be defining a route from point A to Point B along a transit system, moving goods from Point A to Point B along a railroad or waterway system, or defining evacuation routes along roads in case of disasters.

### TranSeg

TranSeg represents a linear section of the physical transportation network designed for human or vehicular movement. A TranSeg may be defined in a variety of ways depending on mode (road, rail, etc) and business application. It is left to the data creator to decide how to segment their transportation system in a manner that supports their organizational functions. A single TranSeg can represent an entire segment between two points, or a separate TranSeg can be defined for each direction of travel. Defining how and where segments are defined is dictated by the need of the application and the dataset being exchanged.

A TranSeg has three required attributes: *status*, *fieldMeasure* and *length* (in addition to the required *identifier* and *lastUpdateDate* attributes inherited from Tranfeature). Example values of *status* include “open to traffic”, “under construction”, “abandoned”, or “proposed”, etc. The attribute fieldMeasure is the length of the segment, as determined in the field, versus the attribute *length* which may different from the field measured length due to differences in calculation (this may be a software-calculated length). Optional attributes for TranSeg include *geometry* and *topology*.

In Figure 3.7.1, note that TranSeg is an <<abstract>> class. That means it has no direct instances; there are no generic transportation segments that correspond to anything in the real world. The other transportations parts: rail (Part 7b), road (Part 7c), transit (part 7d), and water (7e) each have subtypes that are instantiations of the TranSeg abstract class and inherit its attributes. In other words, an implementation of a transportation model could include instances of RoadSeg, RailSeg, ConnectionSeg (transit) or SailingLine (waterways), but an implementation could *not* include a TranSeg. For background about abstract classes and instances, see subchapter 2.2 of this guidance document.

### TranPoint

TranPoints are used to provide topological connections between TranSegs. It is highly recommended that each TranSeg have exactly one start TranPoint and one end TranPoint, as shown in Figure 3.7.2.

Including TranPoints at the start and end of each Transeg will help insure that the resultant set of TranSegs forms a complete coverage of the transportation system without gaps or overlaps. For example, if a roadway transportation network is segmented at all roadway intersections, each TranSeg represents the physical roadway between two intersections and the TranPoints correspond to intersection locations, as shown in Figure 3.7.2. Another possible configuration would be to divide segments at a regular interval. Intersections would still exist along the linear features, but would be represented as TranFeatures instead of TranPoints since they would not define the start and end of each segment. For exchanging datasets without such explicit start and end connectivity, TranPoints can be optional.

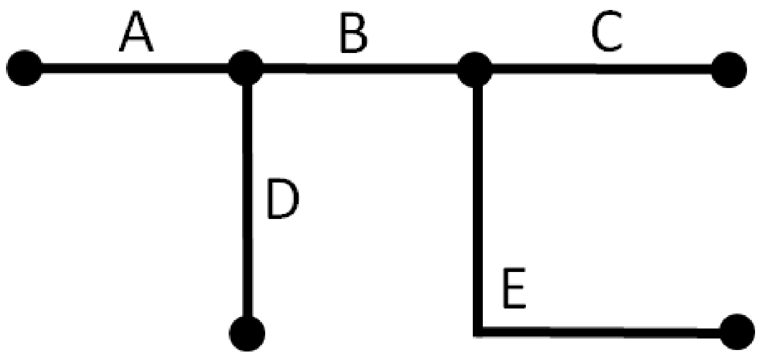


**Figure 3.7.2**
**An example of a segmentation of linear features, where TranPoints (black circles) exist at each start, end, and intersection. This type of segmentation results in five TranSegs (A,B,C,D, and E).**

In addition to providing connectivity between TranSegs, TranPoints also provide direction. Direction is important for railroads, one-way streets, divided highways and many other transportation features. The direction of a TranSeg is determined by its relationship to TranPoints through the "startPoint", "endPoint" roles, as shown in Figure 3.7.3. This type of relationship can implemented several ways. For an example, in a relational database environment, a table could have three columns called SegmentID, StartPoint and End Point. The SegmentID column contains identifiers for each TranSeg, the StartPoint column contains identifiers of the TranPoints that are the startPoints, and the EndPoint column contains the identifiers of the TranPoints that are the endPoints.

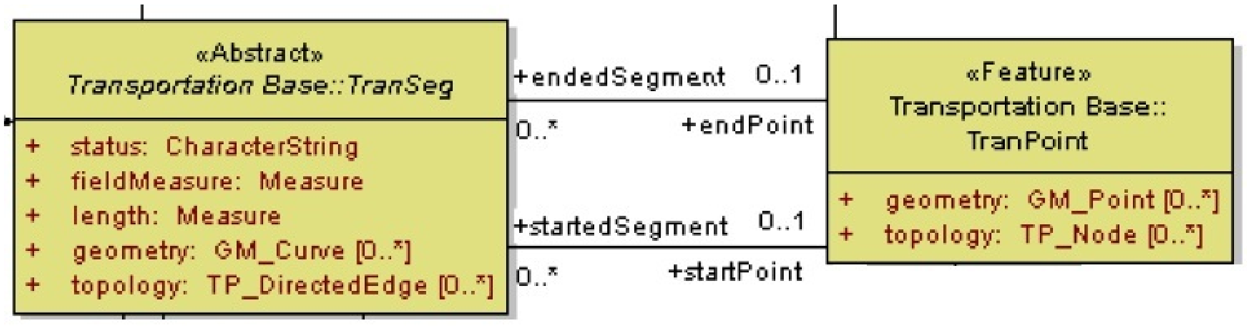

«Abstract»
Transportation Base::TranSeg
+ status: CharacterString
+ fieldMeasure: Measure
+ length: Measure
+ geometry: GM_Curve [0..*]
+ topology: TP_DirectedEdge [0..*]
+endedSegment 0..1
0..* +endPoint
+startedSegment 0..1
0..* +startPoint
«Feature»
Transportation Base::
TranPoint
+ geometry: GM_Point [0..*]
+ topology: TP_Node [0..*]

**Figure 3.7.3**
**Relationships between TranSeg and TranPoint for modeling direction in a transportation network (detail of Figure 3 in Part 7: Transportation Base).**

TranPoints should *not* be used to represent real-world features such as signs, bridges, etc. These types of features are better represented by TranFeatures or by TranFeatureEvents (see subchapter 3.7.2.2). These classes allow for other descriptive attributes, such as sign type or bridge type. They can still be associated with a transportation network. TranPoints do not allow for any descriptive attributes, only for geometry and topology attributes, since they are designed only to start, connect and end TranSegs.

Unlike the abstract class for TranSeg, the TranPoints class is a feature that can be directly instantiated. If a TranPoint is used to connect a RailSeg and a RoadSeg, for the purposes of routing or other network analysis, it could be represented as a single TranPoint rather than as two separate features either a RailPoint or a RoadPoint.

<u>TranPath</u>
The third subtype related to linear features is TranPath. TranPath, as applied in the Rail, Roads, and Transit modal parts of the standard, shall represent how the TranSegs are organized and used such as administrative routes like US 50, or bus or train routes. Because it is a path through the physical transportation system, a TranPath is an ordered collection of one or more, whole or partial, TranSegs it uses. A TranPath will be associated with the identifiers for the Transeg(s) it is connected to.

Since a TranPath is a collection of TranSegs, it inherits any geometry that may be defined by the TranSeg parts that comprise it. Optionally, it may also have its own geometry. For example, the TranSeg geometries may be a more precise representation, whereas the TranPath geometry may be a more generalized representation used for quicker, more efficient network-type analyses.

TranPaths are usually made up of connected TranSegs, but they do not have to be. For instance, a single Tranpath called "Adams Street" may be made up of five segments, two of them west of a river and three of them east of a river, and not connected by a bridge (see Figure 3.7.4).

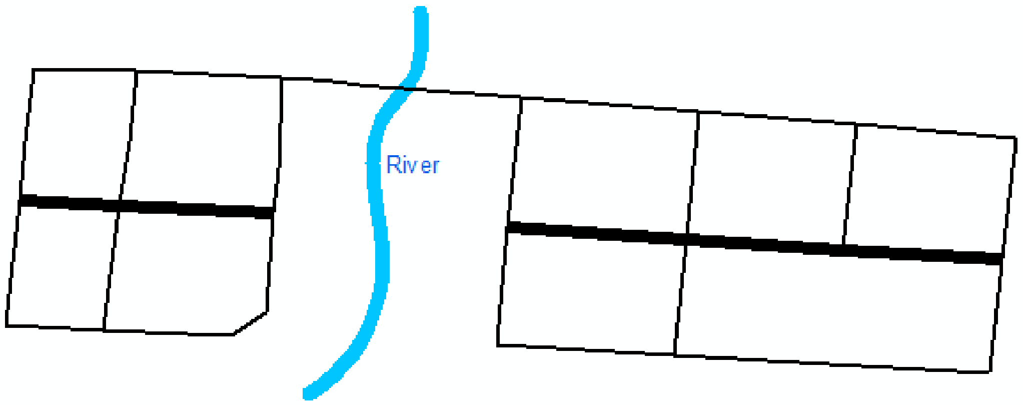


**Figure 3.7.4**
**The five bold segments represent a disconnected TranPath called "Adams Street"**

Another instance of a TranPath may involve a driving path from location A on the east part of Adams Street to location B on the west part of Adams Street. This path would include additional segments, where the driver would turn off Adams Street and travel to another intersection to get to a street that has a bridge to cross the river. After crossing the bridge, they would turn and head back to Adams Street in order to arrive at their destination. In this case, all the segments making up the TranPath are connected (see Figure 3.7.5).

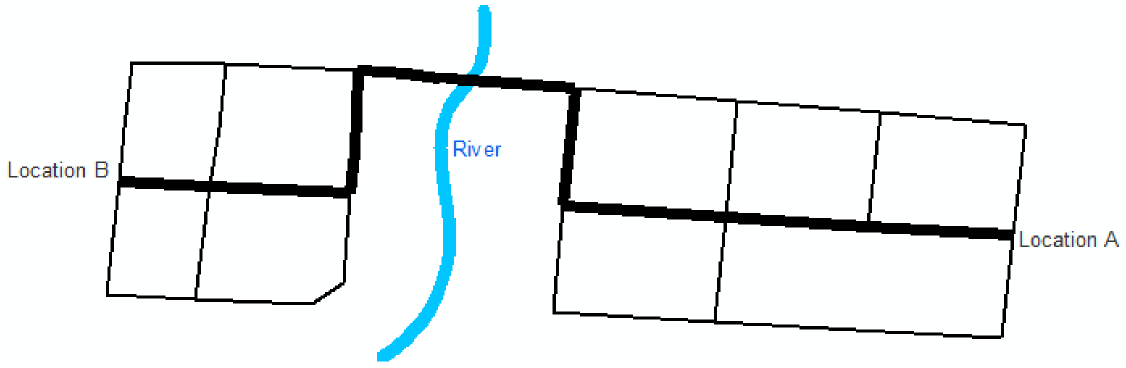


**Figure 3.7.5**
**The eight bold segments represent a connected TranPath called "Driving path from location A to location B"**

A TranPath has one required attribute, *routeNumber* (in addition to the required attributes *identifier* and *lastUpdateDate* inherited from TranFeature). Since it is made up of one or more whole or partial TranSegs, a TranPath will also have the *identifier* and *status* attributes associated with each of its TranSegs.

#### *3.7.2.2 Transportation Event Model*

Events are the mechanism by which attributes or features can be linearly located along either a TranSeg or a TranPath linear feature. The location is specified by a measure, for instance an accident may have occurred at 5.5 miles from the beginning point of a transportation segment or path. Events can also have length associated with them. For instance, a stretch of construction may exist from 10 km to 12 km along a railroad. Events can also optionally include an offset distance, if for instance the accident resulted in a vehicle being found 10 feet to the left of the road (assuming that the road segment or path has direction).

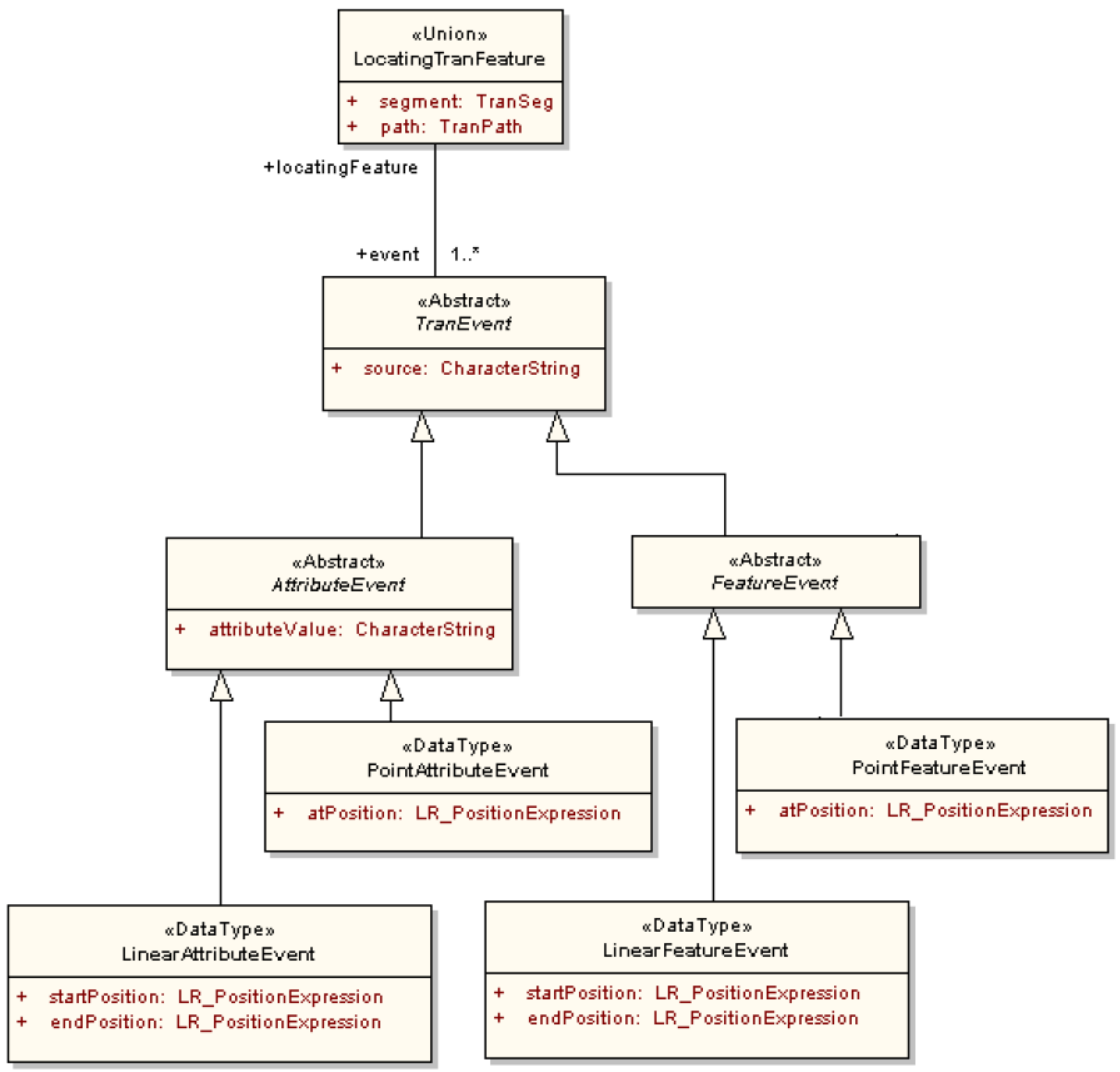

«Union»
LocatingTranFeature
+ segment: TranSeg
+ path: TranPath
+locatingFeature
+event
1..*
«Abstract»
TranEvent
+ source: CharacterString
«Abstract»
AttributeEvent
+ attributeValue: CharacterString
«Abstract»
FeatureEvent
«DataType»
PointAttributeEvent
+ atPosition: LR_PositionExpression
«DataType»
PointFeatureEvent
+ atPosition: LR_PositionExpression
«DataType»
LinearAttributeEvent
+ startPosition: LR_PositionExpression
+ endPosition: LR_PositionExpression
«DataType»
LinearFeatureEvent
+ startPosition: LR_PositionExpression
+ endPosition: LR_PositionExpression

**Figure 3.7.6**
**The Transportation Event model (detail of Figure 7 in Part 7: Transportation Base document).**

At the top of Figure 3.7.6, the class LocatingTranFeature is a <<union>> class, which is a data type consisting of a set of alternatives, only one of which can be used for a particular instance. This means that a single TranEvent must be located on either a TranPath, or a TranSeg, but not both.

A TranEvent is an <<abstract>> event, and both of its subtypes, AttributeEvent and FeatureEvent are also <<abstract classes>>, passing on their attributes to the four <<data type>> classes at the lowest level of the diagram, which can be instantiated.  Another way of looking at this data model is that in actual transportation implementations, there are four types of events that can be located on either TranPaths or TranSegs. These four types can be divided into two categories: AttributeEvents and FeatureEvents.

## AttributeEvents

AttributeEvents are used in the case where the value of an attribute can change at a single location on a TranSeg or TranPath (a PointAttributeEvent) or along a length with a starting point and ending point along a TranSeg or TranPath (a LinearAttributeEvent).

An example of a LinearAttributeEvent is where a highway (a TranPath) starts out with good pavement condition but then has a stretch of poor pavement quality, it might important to know the location where the poor quality pavement occurs. It may be more efficient to create AttributeEvents to define the locations where the pavement quality changes, rather than to create separate features for each stretch of good quality pavement and poor quality pavement. Figure 3.7.7 shows an AttributeEvent along a highway (a TranPath) where the poor pavement quality is stored as two measures, mile 2 and mile 3.

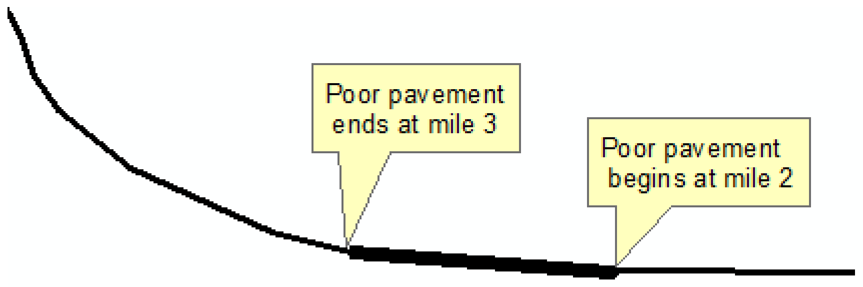


**Figure 3.7.7**
**A LinearAttributeEvent (bold segment) that locates a stretch of poor pavement along a highway (a TranPath).**

Pavement quality is essentially an attribute that will change value at some future point when the pavement is improved, rather than a distinct feature with a need for permanent identifier. Therefore, storing the location of the poor quality pavement as an AttributeEvent is a more efficient representation because it requires storing only two measures (M1 and M2), instead of two or more sets of coordinate pairs (X1, Y1 and X2,Y2 or even more if the segment of poor pavement quality needs more coordinate pairs to define its shape). Since the shape of the underlying TranPath is already defined by the geometry of its TranSeg(s), using the linear locating method means less redundant data.

Other examples of LinearAttributeEvents are a length of a segment or path designated for a particular use or function, such as construction or speed limit zone.

AttributeEvents have two required attributes, *source* and *attributeValue*. For the pavement quality example given above, the source might be a value such as “Survey done on X date by X person” and attributeValue might be “poor” or “fair” or “good.” Other requirements are the reference to the TranSeg(s) or TranPath(s) that are used to locate the event, and the measured location(s) which are *startPosition* and *endPosition* for an event with length (see class LinearAttributeEvent in Figure 3.7.6).

Where only a single measure needs to be located, such as a crosswalk where vehicles traveling along a road are required to stop if there are any pedestrians using the crosswalk, the class PointAttributeEvent should be used. The crosswalk isn’t a physical feature, just a designated location, so it is more efficiently represented as an AttributeEvent than as a feature with its own geometry. The class PointAttributeEvent uses *atPosition* to record its location as a single measurement along the TranPath or TranSeg.

<u>FeatureEvents</u>
FeatureEvents are used in the case where attributes have an unchanging or rarely changing value. A single location on a TranSeg or TranPath with constant value is best represented by a PointFeatureEvent. A length with a starting point and ending point along a TranSeg or TranPath with constant value is a LinearAttributeEvent.

An example of a PointFeatureEvent is a street sign or signal. All the properties of the sign or signal, including its geometry, are associated with an instance of a TranFeature, including user-defined attributes describing its type (stop sign, no turn allowed sign, etc) height, and material. These values don't change, or very rarely change if the sign needs to be replaced. The PointFeatureEvent stores the location information tying the feature to the TranSeg or TranPath.

PointFeatureEvents are often used in the case where features have been collected with a GPS for accuracy. Even if a feature has its own geometry and coordinates, it is advantageous to associate it with FeatureEvent to give it a linear location along a segment or path for certain types of network analysis.

An example of LinearFeatureEvent is a guardrail, which is an actual physical entity and may have its own attributes, such as type or date installed. A guardrail LinearFeatureEvent will also likely have an offset associated with it, for instance the rail is located 15 feet to the left of the centerline of the road (assuming that the road segment or path has direction).

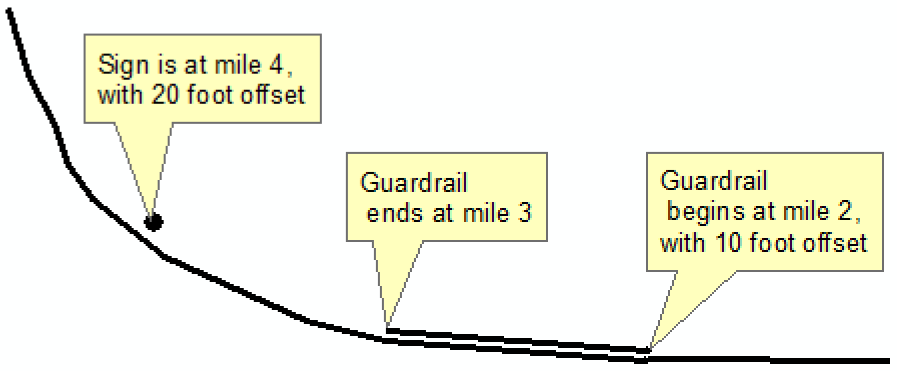

Sign is at mile 4, with 20 foot offset
Guardrail ends at mile 3
Guardrail begins at mile 2, with 10 foot offset

**Figure 3.7.8**
**A sign, represented as a PointFeatureEvent, and a guardrail, represented as a LinearFeatureEvent, both with their own geometry and also linearly referenced to a highway (TranPath) using miles as a linear measurement and feet as an offset measurement.**

Features with area geometries, like a county or some other jurisdiction, are also supported by LinearFeatureEvents. In this case, the LinearFeatureEvent depicts what part of the segment or path is in the county feature.

FeatureEvents have the required *source* attribute and as many other optional attributes as are needed to describe a particular feature, including an optional *geometry* attribute. Other requirements are the reference to the TranSeg(s) or TranPath(s) that are used to locate the linear reference, and the measured location(s). For a PointFeatureEvent, the attribute *atPosition* is the measurement along the TranSeg or TranPath where the PointFeatureEvent is located. For a LinearFeatureEvent, the attributes *startPosition* and *endPosition* provide the measurements to locate the feature event.

The *start*, *end* and *at* positions used to locate an event are specified using a linearly referenced position expression. This expression specifies the linear reference method used to perform the measurement, the linear feature (TranSeg or TranPath) being measured, the measurement along the TranSeg or TranPath, and optionally the measurement laterally offset to either side. For more information about the specific data types used in linear referencing, please see Annex B of Part 7: Transportation Base document. This provides details about the Linear Reference System conceptual model from ISO 19133: Geographic Information – Location base services – Tracking and navigation.

As an example of one type of implementation, all the information needed for a location expression can be pulled from a record (row) in an event table. The event table can be related or linked to the feature table or other file structure that contains instances of TranSegs and/or TranPaths.

### *3.7.2.3 Equivalencies*

As mentioned earlier, "equivalency" is the term given to the process of equating transportation segments, points or paths from disparate databases. Annex A of the Part 7: Transportation Base document provides an informative example of a road, represented three different ways by different agencies. Different databases may have varying positional accuracies, different linear reference methods (LRMs), or different schemes for partitioning (segmenting) the transportation network.

Figure 7.3.10 shows how an equivalence is modeled, in this case for TranPath, though equivalencies are also possible for TranSegs and TranPoints. Equivalency is optional, as indicated by the 0..* notations. The arrow that points back to the TranPath means that one instance of a TranPath can have a relationship with one or more other instances of a TranPath, and this relationship will express the equivalency between the instances.

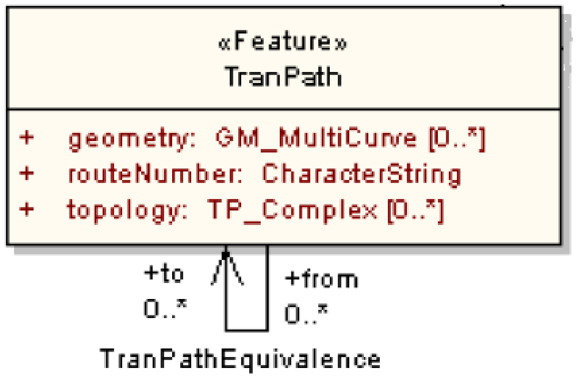

«Feature»
TranPath
+ geometry: GM_MultiCurve [0..*]
+ routeNumber: CharacterString
+ topology: TP_Complex [0..*]
+to
0..*
+from
0..*
TranPathEquivalence

**Figure 3.7.9**
**TranPath class with equivalency relationship (detail from Figure 5 in Part 7: Transportation Base)**

A potential implementation of an equivalency is a relational table, where one column stores identifiers for the source feature, another column stores identifiers for a destination feature, and another column stories the equivalency description. An example of a equivalency description is “source RoadSeg is equivalent to the first 25.46% of the destination RoadSeg”.

### 3.7.3 Part 7b: Transportation – Rail

#### *3.7.3.1 Overview*

The Transportation - Rail part is one of the five modes of transportation systems: air, rail, railroad, transit, and water, defined as pertinent for Framework data. This part of the Framework Data Content Standard attempts to accommodate the principal aspects of rail transportation including geographic locations, interconnectedness, and characteristics. The rail system includes physical and non-physical components representing the rail mode of travel that allow the movement of goods and people between locations. It also includes the supporting infrastructure necessary for rail operations and maintenance. Readers unfamiliar with terminology associated with railways may find it helpful to review features and their descriptions listed in subchapter 3.7.3.4 of this document.

Through methods outlined in Part 7: Transportation Base document (summarized in subchapter 3.7.1 of this document), the rail system can be integrated with other modes of transportation.

The Part 7b: Transportation - Rail document, section 8.1 includes a description of how the Standard handles the differences between the engineering and operation requirements of a rail system. The engineering view requires a very precise representation of all the tracks, switches, signals, platforms and other infrastructure required by a rail system, for proper functioning. The operation view is more concerned with the connectivity and capacity of the rail system required for organizing the movement and timing of passenger or freight trains: e.g. what routes the trains take, how often the trains run, how much can be transported. From the operational point of view applications are not necessarily required to capture each track individually. Instead, a collection of adjacent tracks is represented as a centerline. See section 8.1 of Part 7b for more details and a figure.

The following subchapters summarize how the rail system model is organized, using:

- A segmentation model, derived from the Transportation Base model that defines segments and their associated geometries and topology. The segmentation model includes a class specific to the rail system called a “RailTrack”
- An event model which defines a method to model attributes that may have values that may change along the length of a segment or path and to linearly locate features along segments and paths
- A facilities or administrative areas package defining the important features in a rail system

#### *3.7.3.2 Rail segmentation model*

This subchapter summarizes information from sections 8.1 and 8.2 in Part 7b: Transportation – Rail.

The rail network is the set of rail features and their topological relationships which together define all possible movements through the rail system. The Rail segmentation model provides defines the representation of the physical segments of the rail network (rail segments), their connectivity (rail points), and their usage (rail paths).

The Rail system utilizes subtypes of the Transportation base model (see subchapter 3.7.2.1). For instance, the TranSeg class has a RailSeg subtype, the TranPoint class a subtype, RailPoint, and the TranPath class has a subtype, RailPath. In addition, the RailSeg class as one subtype, RailTrack.

The RailSeg, RailPoint and RailPath classes inherit the attributes defined by their parent Transportation classes, as well as the attributes inherited from the highest parent class, Feature (one require attribute: *identifier*). The RailSeg and RailPath classes include some additional attributes. Both the UML diagram for the Rail segmentation model (Figure 3 in Part 7b) and its matching data dictionary (Table 1 in Part 7b) include descriptions of the Transportation Base attributes and the Rail attributes.

A RailSeg represents a linear section of the physical rail network designed for the movement of trains. Based on whether the rail system is implemented from an operational or an engineering point of view, RailSegs may be defined different ways. For example, a single RailSeg can represent more than one track for an operational implementation, or RailSegs can be defined for each track, for an engineering implementation. RailSegs can have an integer-valued attribute that identifies the number of tracks it represents. Since the number of tracks can vary along the length of a RailSeg, it is more properly represented as a RailLinearAttributeEvent (see subchapter 3.7.3.4).

Like their parent class, TranSeg, RailSegs are not required to have geometry; they can be represented with topology instead for different sorts of applications.

In addition to its inherited TranSeg attributes (required attributes *status*, *fieldMeasure* and *length*), RailSeg has one other required attribute, *isInService*, is Boolean (either true: in service, or false: not in service). It has several optional attributes, such as *owner*, *inServiceDate*, *outOfServiceDate*, *isAlignment*, *startingMileage*, *endingMileage* and *isFerry*.

A RailPoint is a location along the rail network that has significance either for starting or ending a RailSeg. As described in the Transportation Base document, as subtypes of TranPoints, RailPoints are used to provide connectivity between RailSegs, and can also determine directionality in a rail network. RailPoints should not be used to represent other types of rail features, such as signals, switches, etc.

A RailPath represents a route through the physical rail network. It is an ordered list of one or more whole or partial RailSegs. RailPath can be used to represent a connection between an origin and destination. An example of this is an Amtrak Route between Union Station in Washington, D.C. and Penn Station in New York. In addition to the attributes it inherits from TranPath (required attributes are *routeNumber* and the ordered list of RailSegs that make up the RailPath), the RailPath class has three additional required attributes: *isActive* (boolean), *startingPlace* and *endingPlace* (descriptions of the origin and destination). Optional attributes are *name* and *operator*.

RailTrack is a subtype of RailSeg and is specific to the rail model. It is used to provide a more specific engineering-level description of tracks. RailTrack is used to represent the centerline of each pair of rails. In the case of a monorail, RailTrack is at the centerline of the monorail. RailTrack inherits all the attributes of its parent classes, as shown in Figure 3.7.10, and has two additional attributes, *trackType* and *operator*, which are optional. The attribute *trackType* uses a codelist called TrackType, which includes values such as mainTrack, branchLine, interlocking, yard, siding. Recall that codelists are not exhaustive and additional values may be added.

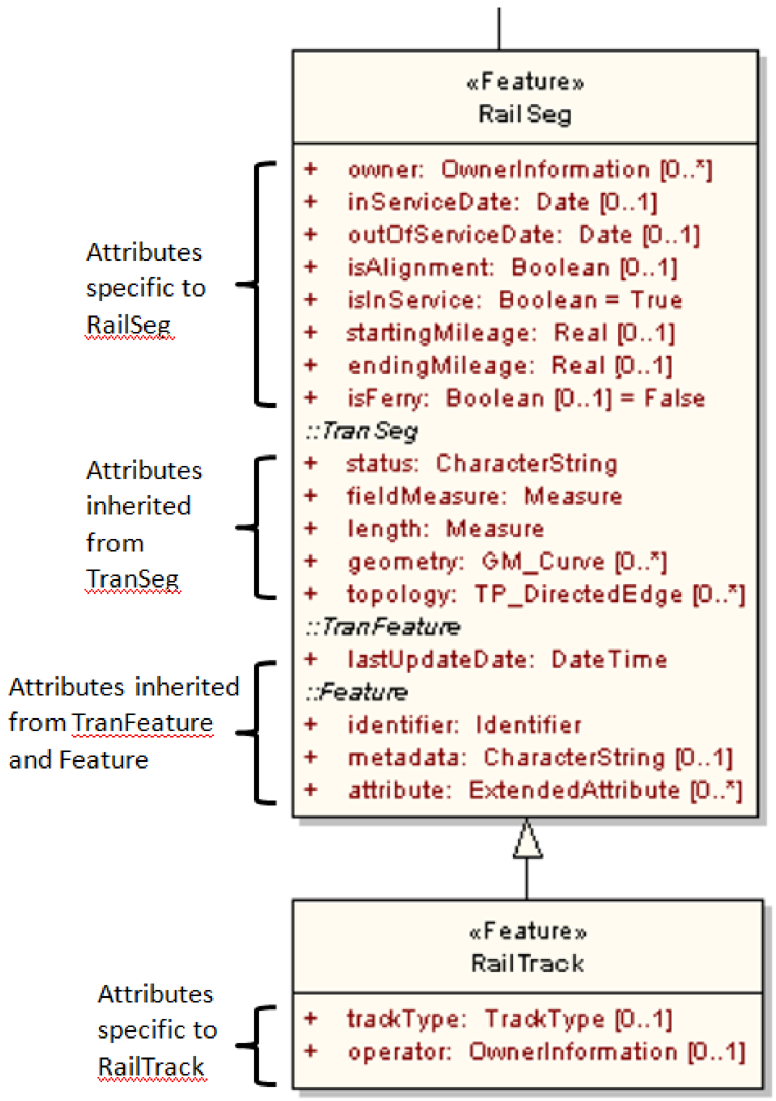

«Feature»
RailSeg
+ owner: OwnerInformation [0..*]
+ inServiceDate: Date [0..1]
+ outOfServiceDate: Date [0..1]
+ isAlignment: Boolean [0..1]
+ isInService: Boolean = True
+ startingMileage: Real [0..1]
+ endingMileage: Real [0..1]
+ isFerry: Boolean [0..1] = False
::TranSeg
+ status: CharacterString
+ fieldMeasure: Measure
+ length: Measure
+ geometry: GM_Curve [0..*]
+ topology: TP_DirectedEdge [0..*]
::TranFeature
+ lastUpdateDate: DateTime
::Feature
+ identifier: Identifier
+ metadata: CharacterString [0..1]
+ attribute: ExtendedAttribute [0..*]
«Feature»
RailTrack
+ trackType: TrackType [0..1]
+ operator: OwnerInformation [0..1]
Attributes specific to RailSeg
Attributes inherited from TranSeg
Attributes inherited from TranFeature and Feature
Attributes specific to RailTrack

**Figure 3.7.10**
**RailSeg and its subtype, RailTrack (detail from Figure 6 from Part 7b: Transportation - Rail)**

### *3.7.3.3 Rail event model*

This subchapter summarizes information from sections 8.3 and 8.4 in Part 7b: Transportation – Rail.

Transportation events are the mechanism by which attributes or entities can be linearly located along either a RailSeg or RailPath feature. Refer to the transportation event model in the Transportation Base for more details, or subchapter 3.7.2.3 in this guidance document.

RailPointAttributeEvents have only one location along a RailSeg or RailPath. They can be either a sign, switch, signal or tower, as defined by the required attribute *pointEventType*. They are allowed only one value, a CharacterString that could describe, for instance, the signal type. Another required attribute is *atPosition*, which is a measurement along either a RailSeg or a RailPath that defines the location. The location can also include an offset distance, for instance if the signal is 10 feet to the left of a RailSeg.

RailLinearAttributeEvents are attributes that vary along lengths of a RailSeg or RailPath, which is defined by a *startPosition* and *EndPosition*. The name of the attribute is specified by the linearEventType property. The value of the segment or path attribute is specified through the attributeValue property. An example of a LinearAttributeEvent in the rail system is a speed restriction: for instance, for from its startPosition at 0 miles to 4 miles, the RailSeg has a speed restriction of 30 mph; from 4 miles on another LinearAttributeEvent is defined for an increased speed. A LinearAttributeEvent also requires that the number of tracks be reported (recall that from an operational point of view, a RailSeg or RailPath can actually be made up of more than one track).

RailFeatureEvents are another type of event, used when it is advantageous to tie an actual feature with its own geometry and attributes, to a location along a RailSeg or RailPath. The next subchapter, 3.7.3.4 provides a list of different rail-related features, all of which carry their own specific set of attributes, and usually their own geometry or topology. For instance, a rail platform may have its own geometry, with associated length or even area, and its own attributes such a name or date constructed. This feature can be associated with a RailFeatureLinearEvent in order to give it a location along a RailSeg or RailPath. Since features can have their own geometry, it also may be more accurate to represent these features for instance with GPS coordinates, rather than just as locations along a RailSeg. However, the RailPointAttributeEvent allows the point features to also be associated with a location along a RailSeg or RailPath for network-type applications, such as “how many railroad crossings exist from mile 0 to mile 100 of a RailPath”.

#### *3.7.3.4 Rail facilities and administrative areas model*

This subchapter summarizes information from sections 8.5 in Part 7b: Transportation – Rail.

This subchapter deals with the supporting infrastructure that is used by the rail industry to conduct ongoing rail operations. These include designated areas for aggregating rail stock, shipping facilities, regulatory signage and signals, and other facilities necessary for the safe and efficient operation of the rail industry.

Each of the feature listed in this subchapter have their own set of attributes, and also inherit attributes from Feature (Part 0: Base document, with one required attribute, *identifier*), TranFeature (Part 7: Transportation Base document, with one required attribute, *lastUpdateDate*) and from the RailFacilityOrAdminFeature class (optional attribute, *owner*). All of these subtypes also include optional associations to RailFeatureEvents (either point or linear, or sometimes both options), so that these features can be located along RailSegs or RailPaths.

- RailStation: named location where railroad or non-railroad revenue and/or operating business occurs; it does not necessarily have to be a building
- RailCommunicationTower: railroad building or structure typically higher than its diameter and high relative to its surroundings that may stand apart, or be attached to a larger structure, that may be fully walled in or of skeleton framework and used for communications
- RailPlatform: raised horizontal platform for embarking to or disembarking from trains
- RailWaysideDetectionDevice: Piece of equipment or a mechanism, adjacent to the railroad, designed to detect characteristics of train movement
- RailYard: system of tracks for storage and maintenance of trains

- RailBridge: structure carrying a railway over a depression or obstacle
- RailControlPoint: location of a track signal or other marker with which dispatchers can specify when controlling trains
- RailFuelingFacility: structure established to transfer fuel onto locomotives
- RailLinearOccupancy: public utility or utilities that occupy land adjacent to the railroad
- RailSignal: A device that indicates to the driver of a train information about the line ahead
- RailSwitch: mechanism used to transfer railcars from one set of tracks to another
- RailTransportationCrossing: intersection between the railroad and another railroad or roadway
- RailUtilityCrossing: intersection between the railroad and public utility equipment

Please refer to the data dictionary associated with a particular feature in section 8.5 of Part 7b: Transportation – Rail for more details about required and optional attributes and their definitions. Section 8.6 includes codelists and enumerations associated with the domains of some of these attributes.

### 3.7.4 Part 7c: Transportation – Road

#### *3.7.4.1 Overview*

The Transportation - Road part is one of the five modes of transportation systems: air, rail, railroad, transit, and water, defined as pertinent for Framework data. This part of the Framework Data Content Standard attempts to accommodate the principal aspects of road transportation including geographic locations, interconnectedness, and characteristics. It is designed to accommodate data associated with the complete road system at all levels of service and all functional classes that may be defined by a data-providing agency. It also accommodates assets associated with roads that are typically used for navigation, safety, and measurement.

The content includes optional elements that will support the development of specialized networks for routing applications, but this level of information is not a requirement.

The content includes optional elements that will support linear referencing systems (LRS). The use of LRS is not added simply to support the requirements of departments of transportation; LRS is used as a technique to transfer road information between systems in a simple, flexible data structure that does not impose a specific segmentation scheme on the data being exchanged. LRS is used in this part of the standard to support the exchange of asset information such as sign locations and pavement condition, as well as to support the placement of transportation statistics such as traffic counts or accident data along the roads, or the number of lanes, or speed limits.

Through methods outlined in Part 7: Transportation Base document (summarized in subchapter 3.7.1 of this document), the road system can be integrated with other modes of transportation.

The following subchapters summarize how the road system model is organized, using:

- A segmentation model, derived from the Transportation Base model that defines segments and their associated geometries and topology.
- An event model, which defines a method to model attributes that may have values that change from one part of a segment to another and to linearly locate features along road segments or paths. In order to linearly locate features, a linear reference model is used, which is described in Annex B of part 7: Transportation Base.

#### *3.7.4.2 Road segmentation model*

This subchapter summarizes information from section 7.1 in Part 7c: Transportation – Road.

The road network is the set of road features and their topological relationships which together define all possible movements through the road system. The road segmentation model defines the representation of the physical segments of the road network (road segments), their connectivity (road points), and their usage (road paths).

The Road segmentation model utilizes subtypes of the Transportation base model (see subchapter 3.7.2.1). The TranSeg class has a RoadSeg subtype, the TranPoint class a subtype, RoadPoint, and the TranPath class has a subtype, RoadPath.

The RoadSeg, RoadPoint and RoadPath classes inherit the attributes defined by their parent Transportation classes, as well as the attributes inherited from the highest parent class, Feature (one require attribute: *identifier*). The RoadSeg and RoadPoint classes include one additional attribute each. Both the UML diagram for the Segmentation model (Figure 1 in Part 7c) and its matching data dictionary (Table 1 in Part 7c) include descriptions of the Transportation Base attributes and the Road attributes.

The road network can be broken up into segments called RoadSegs. RoadSegs represent individual pieces of the physical road network, such as that part of Main Street which exists between First Avenue and Second Avenue. It is important to note that a RoadSeg does not necessarily have to be an entire road. It could be a single lane or carriageway of a road (roads that have physical divisions, usually between traffic going in opposite directions, are said to have dual carriageways).

Furthermore, a segment is not defined as a line on a map, but as a segment of physical road, of which the beginning, end, and length are determined by transportation agencies based on their business needs. The agencies determine where the junctions of segments are placed.

Like their parent class, TranSeg, RoadSegs are not required to have geometry; they can be represented with topology instead for different sorts of applications.

In addition to its inherited TranSeg attributes (required attributes *status*, *fieldMeasure* and *length*), RoadSeg has one other required attribute, *isAnchorSection*, which is Boolean (default is false). An anchor section represents a section of road between two known and recoverable locations, that is to say, anchor points. An anchor point represents a physical location in the field that can be unambiguously described so that it can be clearly located in the real world using the point description. An anchor point is a link between the computer representation of the road system and the real world. Because anchor sections have an anchor point at their beginning and end, they are used to state the official length of a road segment. The function of anchor sections is to support the collection of data by providing an "all distances measured on this piece of road shall add up to this length" checksum.

RoadPoints serve to connect two RoadSegs.  RoadPoints also determine directionality in a rail network. RailPoints should not be used to represent other types of road features, such as traffic signals, stop signs, etc.  RoadPoint has one required attribute, *isAnchorPoint*, which is Boolean (default is false).

RoadPaths prescribe a usage of part of the road network, such as Route 66 or Washington Avenue. They represent a path through a set of whole or partial RoadSegs.

Several implementation examples of RoadSegs and RoadPaths, including figures and tables, are provided in Annex A of Part 7c: Transportation – Road.

#### *3.7.4.3 Road event model*

This subchapter summarizes information from section 7.3 in Part 7c: Transportation – Road.

Transportation events are the mechanism by which attributes or entities can be linearly located along either a RoadSeg or RoadPath feature. Refer to the transportation event model in the Transportation Base for more details, or subchapter 3.7.2.3 in this guidance document.

RoadPointAttributeEvents have only one location along a RoadSeg or RoadPath. A code list of RoadPointEventType values is supplied, which includes tollbooth, tollCharge, maxElevation, sign, and pass. These values originate from a codelist for RoadPointEventType, the codelist is not an exhaustive list; other values can be added. Another required attribute is *atPosition*, which is a measurement along either a RoadSeg or a RoadPath that defines the location. The location can also include an offset distance, for instance if a sign is 10 feet to the left of a RoadSeg.

An example of a RoadPointAttributeEvent is a stop sign along a road. The RoadPointEventType value is "sign". An *attributeValue* of "stop" specifies the type of sign. If more information is needed about the sign, the sign shall instead be represented as a feature and then linearly located with a RoadPointFeatureEvent.

RoadPointAttributeEvents can also be used to specify where something like a pedestrian crosswalk crosses the segment or path.

RoadLinearAttributeEvents are attributes that vary along lengths of a RoadSeg or RoadPath, which is defined by a *startPosition* and *EndPosition*. A code list of RoadLinearEventType values is supplied, which includes many different options (see section 7.4 in Part 7c: Transportation – Road for the full codelist and descriptions of values. Additional values can be added, since codelists are not exhaustive.)

An example of a RoadLinearAttributeEvent is the speed limit of a road. The RoadLinearEventType value is "speedRestriction". An attributeValue of 55 MPH might apply for only part of the road segment, delineated by "start" and "end" positions along the road segment. RoadLinearAttributeEvents have no geometry of their own but instead inherit any geometry which may have been defined for the segment or path to which they apply.

RoadFeatureEvents are another type of event, used when it is advantageous to tie an actual feature with its own geometry and attributes, to a location along a RoadSeg or RoadPath. For instance, a guardrail platform may have its own geometry, with associated length, and its own attributes such as material type or date constructed. This feature can be associated with a RoadFeatureLinearEvent in order to give it a location along a RoadSeg or RoadPath. Since features can have their own geometry, it also may be more accurate to represent features like a tollbooth, for instance, with GPS coordinates, rather than just as locations along a RoadSeg. However, the RoadPointAttributeEvent allows the point features to also be associated with a location along a RoadSeg or RoadPath for network-type applications, such as "how many tollbooths exist from mile 0 to mile 100 of a RoadPath".

More detailed implementation examples of speed limits (linear event) and stop sign (point event), including figures and tables, are provided in Annex A of Part 7c: Transportation – Road.

### 3.7.5 Part 7d: Transportation – Transit

#### *3.7.5.1 Overview*

The Transportation - Transit part is one of the five modes of transportation systems: air, rail, railroad, transit, and water, defined as pertinent for Framework data. Transit data in this part refers to the most broadly used elements of the public transportation system and is not representative of an all-inclusive transit model. The transit system primarily represents the bus mode of travel, though subsequent versions of this part of the standard will include rail transit (e.g. subway, light rail). The data models currently cover many aspects of transit including stops, interconnections, facilities, routes, and temporal elements aspects such as trips and arrival and departure times.

Through methods outlined in Part 7: Transportation Base document (summarized in subchapter 3.7.1 of this document), the transit system can be integrated with other modes of transportation.

Annex A of Part 7d: Transportation – Transit includes an informative example or "use case" of a trip itinerary application. For instance, a customer requests a trip itinerary where they specify an origin, a destination, time/date of travel, and other preferences such as cost, mode/carrier, and number of transfers. A Trip Planning System, using data conforming to the Framework Transit System models, processes the request and generates a trip plan, including routes, schedules and necessary fares. It may also check for possible reroutes due to construction or other events. This annex lists the data requirements and shows how the data requirements may be mapped to elements of the standard transit system model. It includes many trip-related considerations such as availability of parking lots, walking distances, accessibility for handicapped travelers, wait times, amenities (bathrooms, benches, etc), and landmarks or points of interest along the route.

Annex B provides a use case of a public transit stop inventory. Quite often different transit agencies operate in overlapping areas, or have complementary services, so there is a great need to share information. This example shows how information related to transit stops is important to many different applications (e.g. amenities located near the stops, plans for increased or decreased service along stops, analysis of emergency operations). This annex also provides a sample database structure showing how information can be linked to standardized transit stop attributes.

Annex C provides a use case for unplanned re-routing due to potential roadside accidents, weather conditions or other emergencies. Such events will affect arrival and departure times, frequency of service, and accessibility of certain transit stops. This annex shows the data requirements for handling a re-routing and re-scheduling application, mapped to the Framework transit data model.

Annex D described a potential approach to transmitting address information relevant to transit system operations. Addresses are important to many transit applications, such as itinerary planning and facility management, and it is often necessary to locate addresses along the transportation system (rail, road) used by a transit system. This annex shows how the draft FGDC Street Address Data Standard can be integrated with Part 7: Transportation base model.

Section 6 of Part 7d: Transportation – Transit discusses the requirements of the transit system data models. This section begins with a UML class diagram showing all Transit classes and their relationships to each other (Figure 1 in Part 7d). For simplicity, this guidance document focuses on subparts of this model. The following subchapters summarize the following three components of the overall transit system model:

- Relationship of the transit system to other transportation features: transit systems usually depend on roads (e.g. buses) or rails (e.g. subways, light rail). Therefore, the transit system model must also rely on other Framework Transportation data models, described in Part 7: Transportation Base and in more detail in Part 7b: Transportation – Rail and Part 7c: Transportation – Road.
- Transit features: transit systems have their own specific physical features, in addition to the roads or rails (transit paths) that they operate on. These include transit stops and their related features, timepoints (for associating temporal data such as arrival and departure times). Physical features also include transit clusters (one or more transit stops where transfer to another route is possible) and related features such as facilities and landmarks.
- Other transit data types: information which does not have its own geographic location, but is related to one or more of the features listed above. Includes trips, fares, public transportation vehicles (PTvehicles), as well as the assigned patterns and routes that PTvehicles follow.

#### *3.7.5.2 Relationship of the transit system to other transportation features*

This subchapter refers to information from section 6.6 in Part 7d: Transportation – Transit.

Most public transportation vehicles move on either roads or rails, and therefore the transit system model provides the option to link to road and rail data that conforms to Part 7b and 7c, or optionally the more generic transportation linear network defined in Part 7: Transportation – Base (a possibility in the case where a transit system uses physical features from both rail and road networks).

The transportation segmentation model defines the representation of the physical segments of the road or rail network (segments), their connectivity (points where segments connect), and their usage (paths). For more information about TranSegs and TranPoints, see subchapter 3.7.2.1. TranPaths are collections of TranSegs that describe a usage of part of the network, and TransitPath is the subtype of TranPath that can be used to relate transit systems to road or rail systems. For instance, a TransitPath may be collection of road segments that are part of several bus routes, or a collection of rail segments that are part of a light rail route.

As a subtype of TranPath, TransitPath inherits several attributes, topology, and geometry, which are both optional (see subchapter 3.7.2.1 for a discussion of the merits of topology versus geometry and vice versa). Figure 3.7.11 shows several additional options for representing a TransitPath, in addition to geometry or topology. Some applications may not require any geometry or spatial location at all; simply an ordered collection of TransitStops, which are locations where customers can access the transit system.

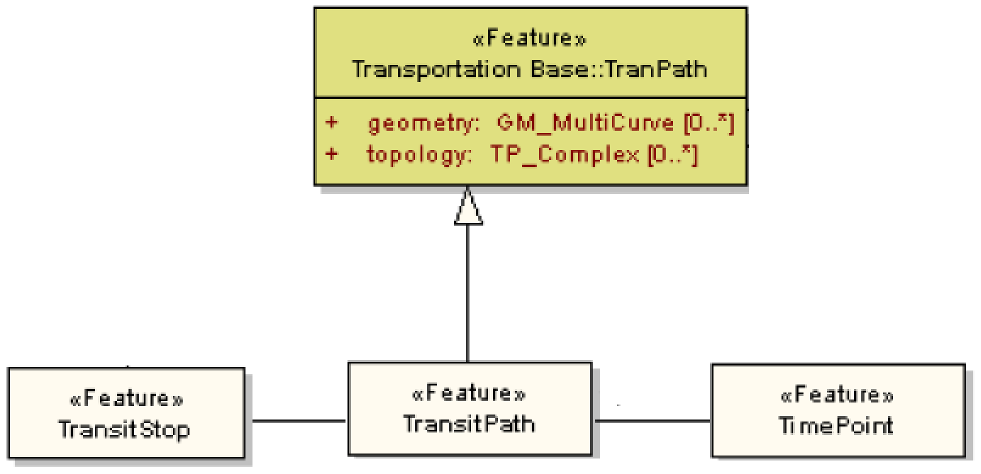

«Feature»
Transportation Base::TranPath
+ geometry: GM_MultiCurve [0..*]
+ topology: TP_Complex [0..*]
«Feature»
TransitStop
«Feature»
TransitPath
«Feature»
TimePoint

**Figure 3.7.11**
**A simplified UML class diagram showing how a TransitPath can be modeled several ways: as an ordered collection of transportation segments (inheriting geometry or topology from TranPath), or as an ordered collection of TransitStops or TimePoints (described more in subchapter 3.7.5.3). The details of these relationships are shown in the class diagram in Figure 3.7.12. These different options reflect the wide variety of business practices in the transit community.**

For instance, as shown in Table 3.7.1, a TransitPath representing a bus route may simply be defined as a list of four stops, ordered in the direction the bus travels. The stops themselves may be defined by a geometry (a point with coordinates) or an address, or some other description.

**Table 3.7.1**
**A TransitPath defined by an ordered collection of TransitStops identified by addresses.**

| Order | TransitStop ID | Address |
|---|---|---|
| 1 | A1 | 412 Adams St |
| 2 | A2 | 640 Adams St |
| 3 | A3 | 700 5th St |
| 4 | A4 | 940 5th St |

However a different type of transit application may require TransitPaths with the geometry of the roads the buses uses in the possible event a bus route needs to be re-routed due to construction or an accident. In the examples shown in Table 3.7.2, the TransitPath is defined by an ordered collection of road segments. In this example, there are no TransitStops explicitly associated with the TransitPath. However, in the example of a relational database, a separate table that stores information aboutTransitStops might also contain a link to identifiers for TransitPaths.

**Table 3.7.2**
**A TransitPath defined by an ordered collection of road segments.**

| Order | RoadSeg Identifier |
|---|---|
| 1 | 456 |
| 2 | 455 |
| 3 | 785 |
| 4 | 786 |

In another type of application, it might be desirable to represent TransitPaths as an ordered collection of TransitStops with a linear reference that locates them along the roads. In Table 3.7.3, the TransitPath is defined by an ordered collection of TransitStops, located by a measure along a RoadSeg.

**Table 3.7.3**
**A TransitPath defined by an ordered collection of TransitStops with a measure that allows them to be referenced linearly along an associated RoadSeg.**

| Order | TransitStop ID | RoadSeg Identifier | Measure, meters |
|---|---|---|---|
| 1 | A1 | 456 | 200 |
| 2 | A2 | 455 | 155 |
| 3 | A3 | 785 | 710 |
| 4 | A4 | 786 | 370 |

Figure 3.7.12 is a more detailed UML class diagram that shows how TransitStops and TimePoints can be located by linear referencing to an underlying transportation network, via the TransitPointFeatureEvent class. For more information about PointFeatureEvents, see subchapter 3.7.2.2.

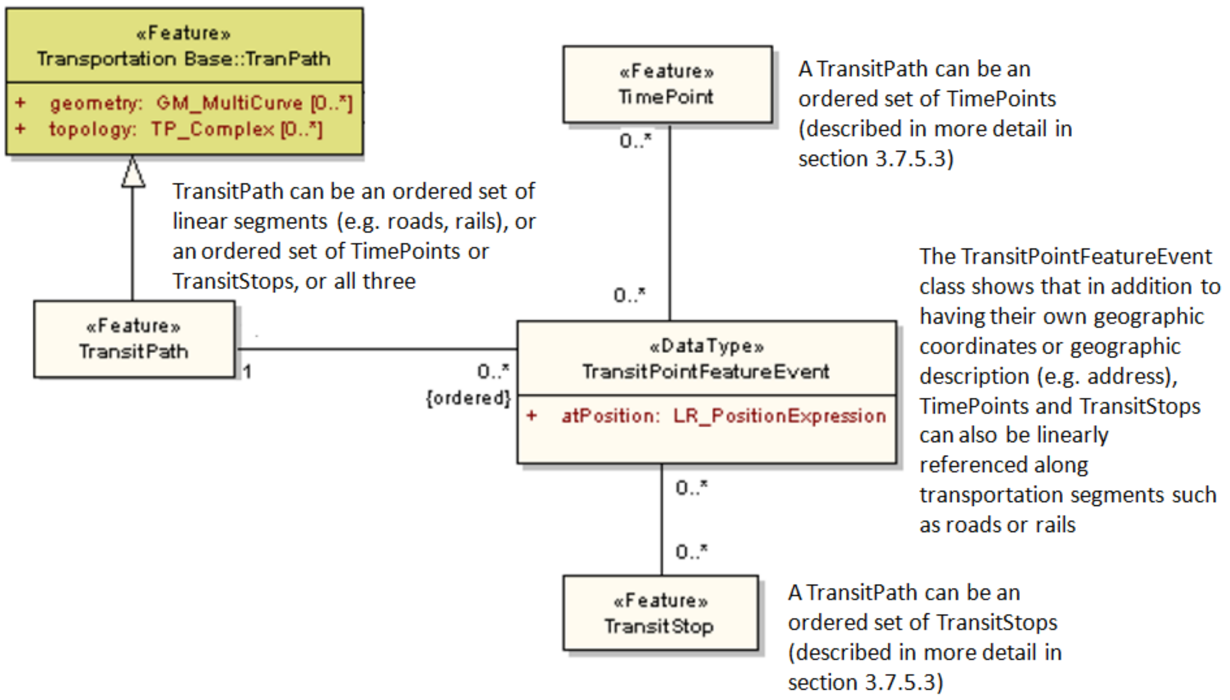

«Feature»
Transportation Base::TranPath
+ geometry: GM_MultiCurve [0..*]
+ topology: TP_Complex [0..*]
«Feature»
TimePoint
A TransitPath can be an ordered set of TimePoints (described in more detail in section 3.7.5.3)
0..*
TransitPath can be an ordered set of linear segments (e.g. roads, rails), or an ordered set of TimePoints or TransitStops, or all three
0..*
The TransitPointFeatureEvent class shows that in addition to having their own geographic coordinates or geographic description (e.g. address), TimePoints and TransitStops can also be linearly referenced along transportation segments such as roads or rails
«Feature»
TransitPath
1
0..*
{ordered}
«DataType»
TransitPointFeatureEvent
+ atPosition: LR_PositionExpression
0..*
0..*
«Feature»
TransitStop
A TransitPath can be an ordered set of TransitStops (described in more detail in section 3.7.5.3)

**Figure 3.7.12**
**A detail of the complete TransitPath data model from Figure 5 in Part 7d: Transportation – Transit.**

### *3.7.5.3 Transit Features*

TransitStop is a central feature of the transit model because it conveys positional information that represents the key business need of providing service to travelers (e.g. pick-up and drop-off locations). For that reason, TransitStops are essential features in the exchange of transit data. A TransitStop has three mandatory attributes:

- *identifier:* a unique identifier (inherited from the overall Framework feature class).
- *lastUpdateDate:* date/time the feature was last updated (inherited from TranFeature).
- *statusInfo*: choice of three values from StatusType codelist: active, inactive, and obsolete.

The optional attributes for TransitStop include:

- *stopId*: unique identifier for a TransitStop
- *stopOwner*: organization that has jurisdiction over the transit stop. The organization is described using the datatype CI_ResponsibleParty defined in ISO 19115: Geographic Information – Metadata. The prefix CI stands for Contact Information.
- *Heading*: direction of travel or orientation of a transit vehicle

TransitStop also includes optional attributes that allow four different ways to describe the location of the TransitStop:

- *Geometry:* point coordinates, using the datatype GM_Point

- *relativeLocation*: a description such as "100 meter from intersection of Adams St and 5th St." This attribute uses a special datatype that is made up of several additional attributes such as street, distance from an intersection, and so forth.
- *address*: a description such as 1612 Adams St. The address is described using the datatype CI_Address defined in ISO 19115: Geographic Information – Metadata. The prefix CI stands for Contact Information.
- *alongLocation*: a linear reference (e.g. mile 5.2 of Adams St). The linear reference is described using the datatype: LR_PositionExpression, defined in ISO 19133: Geographic Information – Location-based services. The prefix LR standards for Linear Reference.

TransitStops may be associated with several other Transit features, such as TimePoints, TransferClusters and Facilities (described below), and TransitPaths (described in the previous subchapter). See Figure 3 for the complete TransitStop UML diagram and Table 1 for the matching data dictionary in Part 7d: Transportation – Transit.

A TimePoint is a location where Trips are assigned arrival, dwell, or departure time periods. A Trip is a one-way scheduled movement of a transit vehicle between starting and ending TimePoints. In many cases, a TimePoint and a TransitStop are basically the same thing, for instance a TransitStop is where a bus picks up/or drops off passengers at an assigned time. In other cases, a TimePoint may not be associated with a TransitStop– for instance when a bus returns to a garage for storage, maintenance or re-fueling.

A TimePoint has four mandatory attributes, *identifier*, *lastUpdateTime*, *geometry* and *time*. TimePoint's attribute "geometry" must be identified by a point coordinate. Geometry is required, because for accurate and reliable scheduling, an exact location of a transit vehicle should be associated with a scheduled time. For example, a bus may have four TransitStops where it picks up/drops off passengers, but only two TimePoints. It has an assigned departure time (TimePoint 1) at TransitStop A, after which it stops at TransitStops B and C. Then it has an assigned arrival time (TimePoint 2) at TransitStop D (see Figure 3.7.14).

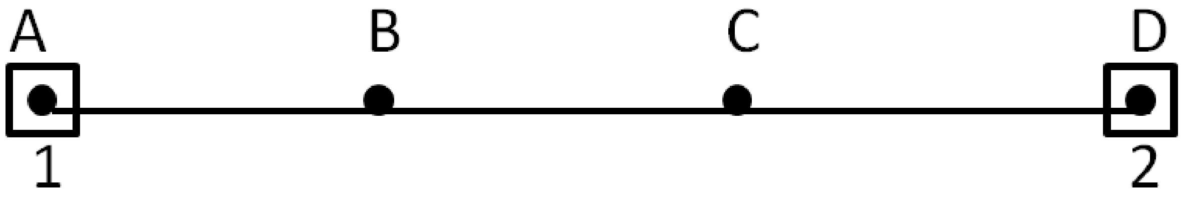


**Figure 3.7.13**
**An example of a bus trip that starts at the square, TimePoint 1, at an assigned departure time. TimePoint 1 is also associated with TransitStop A, where passengers can embark/disembark. The bus travels along a route and stops at TransitStops B, C and D. TransitStop D is also the location of TimePoint 2, which has an assigned arrival time. There are no assigned times for TransitStops B and C.**

In Figure 3.7.13, it can also be seen why TransitStops are not required to have geometry. At TransitStop A and D, the TransitStop can have a location by its association with the TimePoint locations. TransitStop B and C can (optionally) have their own geometry to show location, or their location may be described in alternative ways such as by relative location, address, or a linear reference.

Another more complicated example of TimePoints is shown in Figure 3.7.14. This is where a Trip includes three TimePoints, two of which are visited twice on the trip. In order to handle situations where TimePoints are visited more than once, another class called TransitTemporalEvent can be used. In Figure 3.7.14, the Trip includes five TransitTemporalEvents.

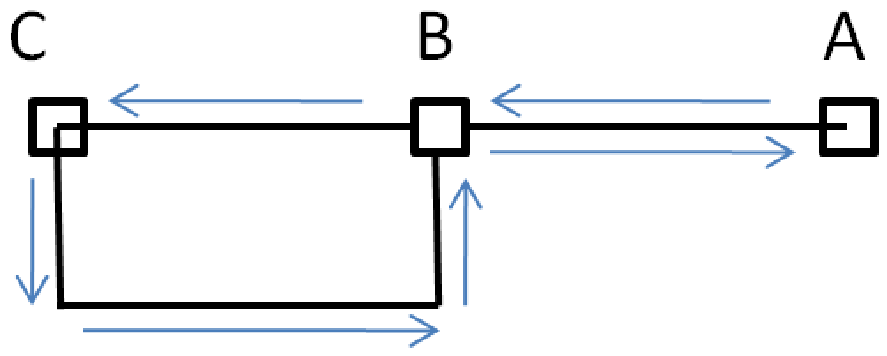


**Figure 3.7.14**
**A bus trip which begins at TimePoint A, progresses to TimePoint B and then C, then makes several turns bringing the bus back to location B again and ends at the same place where the trip began, at location A. Though there are only three specific TimePoint locations, there are actually five different arrival/departure times that are scheduled along this route (TransitTemporalEvents).**

See Figure 6 for the complete TimePoint UML diagram and Table 4 for the matching data dictionary in Part 7d: Transportation – Transit.

A TransferCluster is another type of TransitFeature that is closely related to TransitStop since it is a collection of TransitStops where transit passengers can change routes. It has three mandatory attributes, identifier, lastUpdateDate (inherited) and Name. It has one optional attribute, geometry, which is a polygon. Or its location can be inferred from the locations of the TransitStops that are associated with it. See Figure 11 for the complete TransferCluster UML diagram and Table 9 for the matching data dictionary in Part 7d: Transportation – Transit.

A Facility is a place that is used by a transit agency. Examples of transit facilities include parking locations, rail yards, and administrative offices. *FacilityTypeCode* is a mandatory attribute, and the full codelist for facility types is given in Table 16 of Part 7d: Transportation – Transit. A Facility feature can also have an optional attribute, *address*. One or more TransferClusters, TransitStops, and may optionally be associated with a Facility. For instance, it may be important for customers to know that there is facility with restrooms at some point along a bus trip. See Figure 13 for the complete Facility UML diagram and Table 11 for the matching data dictionary in Part 7d: Transportation – Transit.

An Amenity feature refers to physical elements such as bus shelters, schedule displays, and bike racks that are associated with either a TransitStop or a Facility. It has three required attributes: *name*, *amenityType* (see AmenityType codelist), and *hostingFeature* (the unique identifier for either a TransitStop or a Facility). See Figure 14 for the complete Amenity UML diagram and Table 12 for the matching data dictionary in Part 7d: Transportation – Transit.

A Landmark is feature that represents a point of interest. A Landmark feature has three mandatory attributes, *landmarkName*, *landmarktype*, and *address*. *Geometry* is an optional attribute. Landmarks can be associated with TransferClusters or Facilities. See Figure 12 for the complete LandMark UML diagram and Table 10 for the matching data dictionary in Part 7d: Transportation – Transit.

A ConnectionSeg is a linear path allowing transit riders to move from one TransitStop to another. The segment may be defined as a walking path, bike path, escalator, or other modal connection. ConnectionSeg is a subtype of TranSeg (see subchapter 3.7.2.1) and describes a linear feature that allows a transit rider to move between TransitStops. Mandatory attributes include *distance*, *fromStop* (identifier for a TransitStop), *toStop* (identifier for another TransitStop), and connection *instructions*. An optional attribute, *accessibility*, provides information about whether stairs, narrow sidewalks or other potential obstacles (for customers in wheelchairs, for instance) exist along the connection. See Figure 4 for the complete ConnectionSeg UML diagram and Table 2 for the matching data dictionary in Part 7d: Transportation – Transit.

#### *3.7.5.4 Transit Data Types*

Several Transit data types can be associated with Transit features. Transit data types include Trip, Pattern, PTvehicle, Block, TransitRoute and Fare.

A Trip, as mentioned in the previous subchapter, is a one-way scheduled movement of a transit vehicle between starting and ending TimePoints. Since a Trip is of specific interest to customers, it includes an optional attribute, Fare.

A Pattern is defined as an ordered sequence of TransitStops or TransitPaths that is followed by a transit vehicle in scheduled service. While a Trip is a construct that is useful to costumers, a Pattern is a construct that is useful for transit managers. It has several optional attributes:

- *patternType* (e.g. revenue, training, maintenance – see codelist PatternType for full list)
- *routeDirection* (e.g. inbound, outbound, circular, north, south – see codelist RouteDirectionType)
- *transitServiceType* (e.g. regular, express, feeder, charter, school – see codelist TransitServiceType)
- *timetableVersion* – used if there are multiple time tables associated with a pattern

A Pattern may have one or more TransferClusters and one or more Trips associated with it. See Figure 7 for the Pattern UML diagram and Table 5 for the matching data dictionary in Part 7d: Transportation – Transit.

A TransitRoute is a collection of patterns in revenue service. While patterns are used internally by a transit organization for management, a TransitRoute is what a customer would see, usually in the form of a timetable, which they would use to define a one-way Trip (or Trips). A TransitRoute has three mandatory attributes: *routeNumber*, *name*, and *timeTableHeader*. This last attribute is a summary of publically recognized TimePoints contained in a group of Patterns oriented in the same route direction, and is used to generate timetables. See Figure 8 for the TransitRoute UML diagram and Table 6 for the matching data dictionary in Part 7d: Transportation – Transit.

PTVehicle is a data type that describes a public transportation vehicle, and PTVehicle refers to any public transit conveyance. PTVehicle has one mandatory attribute, *vehicleId*, and several optional attributes such as *vehicleCapacity*, *vehicleType* and other attributes for real-time routing and scheduling status. A

PTVehicle is assigned to one vehicle base or Facility (at a time). See Figure 10 for the PTVehicle UML diagram and Table 8 for the matching data dictionary in Part 7d: Transportation – Transit.

A Block is a sequence of Trips over which a PTVehicle is assigned from pull out time to pull in time. See Figure 9 for the Block UML diagram and Table 7 for the matching data dictionary in Part 7d: Transportation – Transit.

A Fare is a data type that describes the cost for riding a transit vehicle. It has five mandatory attributes:
*fareValue*: a real number
*fareType*: e.g reduced, full-fare, transfer, special
*farePolicy*: flat, distance, zone, timeofDay, etc. (see FarePolicyType codelist for full list of values)
*fromStop:* the identifier of aTransitStop
*toStop*: the identifier of aTransitStop

See Figure 15 for the Fare UML diagram and Table 13 for the matching data dictionary in Part 7d: Transportation – Transit.

### 3.7.6 Part 7e: Transportation – Inland Waterways

#### *3.7.6.1 Overview*

The Transportation – Inland Waterways part is one of the five modes of transportation systems: air, rail, railroad, transit, and water, defined as pertinent for Framework data. Inland waterways refer to navigable rivers that are used for transportation. This effort is derivative of an ongoing program within the U.S. Army Corps of Engineers (USACE), called the Inland Electronic Navigation Charts (IENCs). This part of the Framework Standard describes authoritative data content derived from the IENC.

Through methods outlined in Part 7: Transportation Base document (summarized in subchapter 3.7.1 of this document), the transit system can be integrated with other modes of transportation.

#### *3.7.6.2 Inland Waterways data model*

The inland waterways model has three classes: ProjectDepthArea, MileMarker and SailingLine (see Figure 1 and Table 1 in Part 7e: Transportation – Inland Waterways for a UML diagram and data dictionary of these classes). All three of these classes share a mandatory attribute called *sourceIndicator*, which describes the authoritative source of the data entity.

ProjectDepthArea is a designated navigation area. In addition to the mandatory *sourceIndicator* attribute, this class also has a required attribute, *geometry*, that is defined by a polygon. More specifically, the designated navigation area is within the waterway and is bounded by a depth contour that has minimum depth of 9 feet. ProjectDepthArea is a subtype of TranFeature from Part 7: Transportation Base, and it inherits the mandatory attribute *lastUpdateDate* from this class. It also inherits the mandatory *identifier* attribute from the Feature class identified in Part 0: Base document.

SailingLine describes the recommended transit route, based on sufficient water depth and permitted passage along the waterway. It is a subtype of TranSeg (see subchapter 3.7.2.1), and in addition to the mandatory *sourceIndicator* attribute, it inherits three other mandatory attributes from TranSeg: *status*, *fieldMeasure* and *length*. The attribute *fieldMeasure* is the length of the segment, as determined in the field, versus the attribute *length* which may different from the field measured length due to differences in calculation (this may be a software-calculated length). SailingLine can inherit two optional attributes from TranSeg: *geometry* (GM_curve type) and *topology* (TP_DirectedEdge type). See subchapter 3.7.2.1 for more details about the *geometry* and *topology* attributes.

MileMarker describes the position of a mile marker in or near the waterway to serve as a reference for distance along the waterway. It is a subclass of TranPoint (see subchapter 3.7.2.1) and in addition to the mandatory *sourceIndicator* attribute, it has a mandatory attributes for geometry (GM_Point type). It has two other optional attributes, *information* and *sailingLineId*. The S*ailingLineId* attribute is a unique identifier that associates the MileMarker with a particular SailingLine. The *information* attribute can be any sort of additional information related to the MileMarker.

# 4. Implementation

The Framework Data Content Standard may be implemented in a variety of ways. Implementation can be divided into three basic categories:

1. Creating data content. Producers have several choices for implementation:
    a. Create data from scratch according to the Framework Data Content Standard.
    b. Convert existing data into standardized content.
    c. Keep existing data content intact and use a transformation utility that will convert it as needed to standardized content.
2. Encoding data content: how a producer of geographic information chooses to encode standardized content into a particular file structure such as GML, common GIS software formats such as shapefiles, or common image formats such as tagged image format, and so forth.
3. Sharing data: how a distributor of geographic information chooses to make the data available for other parties to access. This could be as simple as providing files on an FTP site or on some form of deliverable media. Increasingly, OGC Web Mapping Services (WMS), Web Feature Services (WFS) and web coverage services (WCS) are used to serve data on the Internet for viewing and sharing geographic information.

The NSDI Cooperative Agreements Program (CAP) has provided funding for several Framework implementation projects, some of which are available on-line as examples. Training materials and other resources for implementation are available or will be made available on the FDGC website (*www.fgdc.gov*). In particular, separate training documents are available that cover the underlying concepts and deployment of interoperable systems for publishing data compliant with the Framework Standard (Dibner 2010). This chapter provides a brief summary of the detailed implementation guidance that these materials provide.

Names of specific software tools are not provided in this subchapter, as a comparative evaluation of the capabilities of software has not yet been undertaken. The Open Geospatial Consortium (OGC) maintains a list of software resources that meet their specifications at http://www.opengeospatial.org/resource. As of the time of this document's publication, there is not yet a list of software resources that have been compiled specifically for the Framework Data Content Standard.

## 4.1 Creating data content

The UML class diagrams for Framework Data Content Standard are available in XML metadata interchange (XMI) format, which allows the Standard to be brought into various modeling and diagramming tools. Some GIS software can import data models/diagrams to create data templates or schemas, which can simplify the process of data creation. However data producers will still need to determine how to structure data content in terms of a relational model, georelational model, object-oriented model, flat file, and so forth.

Some XML editing software includes mapping or matching utilities which can assist the process of matching an existing data schema to the Framework data content schemas. Data producers will need to look at each element in their existing data and decide which of the Framework data elements it corresponds to. If the existing data is in XML/GML format, after the mapping process has been completed, the XML software programs can convert the data to the new schema. In some cases, the existing data may be missing content required by the Framework Data Content Standard; however the data can still be mapped and converted. In other scenarios, the existing data may include additional content that is not included in the Framework Standard. This additional content can still be mapped across to the optional ExtendedAttribute class that is part of the Framework feature model.

Data producers may also consider programming a "transformation utility" that uses the mapping between existing content and Framework content to automatically transform existing data content to Framework-compliant content. These scripts can be built using virtually any programming environment, include the scripting capabilities built into many GIS software and WFS software. The scripts can be scheduled to run on a regular basis, whenever data is updated, or can be called whenever a request for Framework-compliant data is received. This allows the data to remain in its original schema for internal purposes, but also be made available as Framework-compliant content available for sharing and automated integration with other sources.

The transformation utility may take many forms (Dibner 2010), including:

- A script that runs in the context of your Geographic Information System, and performs an appropriate transformation as data are retrieved.
- An external script or compiled program called as a function by the WFS system - uses a vendor-specific library to pass data back and forth.
- An external process that runs asynchronously in coordination with the WFS - communicates via some interprocess channel (e.g., pipes, messaging, remote procedure call…), also possibly using a vendor-supplied library.
- An independent application that converts the entire native data store and transfers it to an independent, Framework-compliant data store.

## 4.2 Encoding data content

Part 0: Base document of the Framework Standard includes Annex C, an informative section on using Geography Markup Language (GML) for the encoding of geographic information. GML is an open interchange format for geographic data on the Internet, and was designed specifically with the goal of integrating all forms of geographic information, including vector features, coverage features (e.g. images) and sensor data. GML is a specialization of the Extensible Markup Language (XML) developed by the Open Geospatial Consortium (OGC) and is an open source format.

GML contains a rich set of primitives which have been harmonized with the ISO conceptual models used as the basis for describing content in the Framework Data Content Standard. As a result, the data models (e.g. application schemas) in Part 0 through Part 7 have been translated into XML schemas for GML. An XML schema is a description of rules, constraints and allowable data types, and can be used to validate GML data to see if its content conforms to the Framework Standard. For instance, an XML schema document (extension .xsd) can determine if mandatory attributes are present, or if attributes are expressed in the correct data type (e.g. character string or integer), or if attribute values conform to a specific domain (an enumeration of specific allowable values).

If a data provider chooses to make data available in GML, they have the advantage of being able to validate the data to see if it conforms to the Framework XML schemas described in Annex C of Part 0: Base (available at http://frameworkwfs.usgs.gov/framework/schemas/gmlsf1/). Some GIS software or WFS server software can be used to convert geographic information in other formats (e.g. shapefiles) into GML. Both open source and proprietary software products are available.

## 4.3 Sharing data content

OGC standards define interoperable services and encodings that enable the Spatial Web, and have been adopted to provide an interoperable service infrastructure for supporting the NSDI. Examples include WMS (Web Map Service) for delivering images of maps across the internet, WFS (Web Feature Service) for describing and delivering feature data, and WCS (Web Coverage Service) for describing and delivering imagery and other raster data.

The following information is excerpted from “Serving Transportation Data through the NSDI” training materials developed for the FGDC (Dibner, 2010).

Interoperable data distribution services based on OGC standards provide consistent interfaces that enable the creation of client software that can communicate with any of the services. These consistent interfaces, conforming to the notion of a *service oriented architecture* (SOA), are key to interoperability.

Other advantages:

- data can be stored in distributed locations, on many servers throughout the internet
- existing in-house tools and applications remain viable
- standards-compliant client software can communicate with any service instance

The WFS protocol, and OGC specifications in general, do NOT constrain:

- Data format
- Custodial or management policies
- Hardware or software platform

They DO define:

- Lightweight suites of web service operations
- Encodings for data transport (e.g. GML)
- Various means for services to describe their contents and capabilities

The Web Feature Service (WFS) is simply a well-specified interface: it provides a known mechanism for interacting with its underlying system. The underlying system might be a suite of GIS programs, a simple data handling system, or any other system that holds or can distribute geospatial data.

Web Feature Services follow a typical pattern of interaction with client software. The client asks server “what data and operations do you provide?” The server responds with a Capabilities document that describes what it has. The client may choose to view or download the data, or parse the capabilities and feed available choices to the user interface or other programmatic components – such as applications that integrate data from various sources (e.g. multiple servers). The WFS server may directly retrieve the data it serves from a database system, or it may accept data via a functional or object-oriented interface, a remote procedure call, or some other programmatic means. There is a tutorial on using WFS at http://www.ogcnetwork.net/wfstutorial.

## 4.4 Future work

Implementation guidelines in future versions of this guidance document will be improved by more detailed documentation of successful Framework Standard implementations. In addition, implementation efforts will be greatly assisted by a comparative evaluation of the capabilities of software that can be used for building schemas, schema matching, transforming data and serving data.

## Bibliography

Dibner, P.C. 2010. Serving Transportation Data through the NSDI. Powerpoint training materials, FGDC Cooperative Agreement Program. Retrieved December 15, 2010 from http://www.ogcii.org/nsdi_training/.

Federal Geographic Data Committee (FGDC). (1997) *Framework Introduction and Guide.* Washington, DC: Federal Geographic Data Committee.

Frank, S.M., M.F. Goodchild, H.J. Onsrud, and J.K. Pinto. (1996) User requirements for framework geospatial data. *Journal of the Urban and Regional Information Systems Association* 8(2): 38-50.

Hamerlinck, J.D. (2008) Framework data, *In: Encyclopedia of Geographic Information Science,* K.K. Kemp, editor. SAGE Publications: Thousand Oaks, CA: 151-155.

Lorenz, B. (2006) *The Case for Data Interoperability.* Retrieved September 15, 2010 from http://www.facilitiesnet.com/software/article/The-Case-For-Data-Interoperability--4094.

Masser, I. (2005) *GIS Worlds: Creating Spatial Data Infrastructures.* Redlands, CA: ESRI Press.

National Research Council Mapping Science Committee. (1995) *A Data Foundation for the National Spatial Data Infrastructure.* Washington, DC: National Academy Press.

Nebert, D. (Ed.) (2004). *Developing Spatial Data Infrastructures: the SDI Cookbook*, version 2.0. Global Spatial Data Infrastructure. Retrieved September 21, 2010 from http://www.gsdi.org/gsdicookbookindex.

Open Geospatial Consortium (OGC). (2006) T*he OpenGIS Abstract Specification Topic 6: Schema for coverage geometry and functions* (OGC 07-11 version 7.0.0)

Smith, N.S. and D.W. Rhind. (1999) Characteristics and sources of framework data. Chapter 47 *In*: Longley, P.A., M.F. Goodchild, D.J. Maguire, and D.W. Rhind (eds), *Geographical Information Systems,* 2nd edition, New York, John Wiley & Sons: 655-666.

Simsion, G. and Witt, G. (2005) *Data Modeling Essentials,* 3rd edition. Burlington, MA: Morgan Kaufmann.

Ter Bekke, J.H. (1992) *Semantic Data Modeling*. Upper Saddle River, NJ: Prentice Hall.

## List of Hyperlinks (In Order of Appearance)

FGDC Standards Publications < http://www.fgdc.gov/standards/standards_publications/>

Geospatial Interoperability Return on Investment Study <http://www.egy.org/files/ROI_Study.pdf>

Spatial Data Infrastructure (SDI) Cookbook <http://www.gsdi.org/gsdicookbookindex>

Components of the NSDI < http://www.fgdc.gov/components>

FGDC Subcommittee for Cadastral Data < http://www.nationalcad.org/>

National Geodetic Survey < http://www.ngs.noaa.gov>

50 States Initiative < http://www.fgdc.gov/policyandplanning/50states/index_html>

NSDI Training Program < http://www.fgdc.gov/training/nsdi-training-program/online-lessons>

American National Standards Institute < http://www.ansi.org/>

International Committee for Information Technology Standards < http://www.incits.org/>

Office of Management and Budget < http://www.whitehouse.gov/omb/circulars_a119>

National Digital Elevation Program < http://www.ndep.gov/>

MSDI Data: Geodetic Control < http://www.gis.state.mn.us/MSDI/workgroups/geodetic.htm>

Federal Geographic Data Committee < http://www.fgdc.gov/>

Software that meets OGC interoperability specifications at <http://www.opengeospatial.org/resource>

Framework XML Schemas <http://frameworkwfs.usgs.gov/framework/schemas/gmlsf1/>

WFS tutorial < http://www.ogcnetwork.net/wfstutorial>