Introduction to

# PipelineML

---

*An open data interchange standards development initiative of the Open Geospatial Consortium for the Oil & Gas Midstream Pipeline industry*

# *Table of Contents*

# Overview

## *Value Proposition*

Essential to the mission of every oil and gas (midstream) pipeline stakeholder is the need to share and exchange information. Despite this being an industry-wide need, each organization has borne this burden individually. Each company is left to the costly and time-consuming process of developing and defining data management best practices, policies and procedures, and standards (or suffer from a lack thereof). Each organization has had to reinvent the wheel of managing the exchange of large volumes of data associated with new construction, acquisitions, divestures, and changes in operation. The failure to innovate in these areas risk delays in access to the information needed for mission-critical business decision support.

The volume and resolution of pipeline data continues to climb. Requirements placed on operators from regulatory agencies continue to escalate. The time for change is now. The midstream industry is poised for a pivotal shift in how asset data is managed and shared in order to keep pace with market demands. The opportunities and pressures require industry-wide collaboration and innovation. Collectively, industry stakeholders are developing a set of standards that fulfill the following requirements:

- ✓ Expiate timely interchange of information among stakeholders in the asset value-chain
- ✓ Reduce costs in the development and management of industry standards
- ✓ Increase granularity of managed asset data to weld level resolution
- ✓ Streamline regulatory reporting with common shared industry reporting standards
- ✓ Design traceability, verifiability and completeness into the core standards framework
- ✓ Enhance public safety by putting timely, accurate data in the hands of those who manage public awareness
- ✓ Reduce costs and risks of not having current, accurate data readily available to support business and operational drivers

# *Introduction*

PipelineML is an open industry data standard designed to help midstream pipeline operators share information about their assets and the various construction and maintenance activities performed on them. The following are some common reasons why pipeline asset information needs to be shared:

- ✓ **Service Provider Support** – Most operators engage the services of numerous vendors to conduct maintenance and construction on their assets. This requires information to be shared between the operator and the service provider as to the exact location and nature of the work to be performed. Once the work is completed, the vendor must submit detailed information on the work conducted. Each operator and vendor uses different standards, media and formats to share this information. As a result, this process tends to require a person to interpret its meaning and rekey it into their various data management systems. This introduces unnecessary time and financial overhead. Developing an open industry data interchange standard can make this exchange fast and economical while significantly reducing potential errors.

- ✓ **Internal Availability** – Operators need to share asset data between internal departments over the lifecycle of those assets (design, parts acquisition, construction, line lowering, surveys, operations, line location, public awareness, observations and measurements, integrity, rehabilitation, divestiture, etc.). Typically, each department uses different software and systems to manage their data, activities, and decisions. When information is shared between departments, it typically must undergo extensive transformation, remapping and sometimes manual recoding. This can lead to loss of data and introduce errors. It is also costly in terms of time and money. As a result of these barriers, information may not be shared across departments as often as could be beneficial to the organization as a whole. All these factors can be mitigated through the use of a common data interchange standard.

- ✓ **Regulatory Submissions** – Operators must submit information about certain assets and activities to regulatory agencies on an annual basis. Since there are no fully open industry data interchange standards, each agency requests submissions in different formats and specifications. As a result, operators must invest significant efforts to transform that data into numerous formats to meet the requirements of each agency. This situation could be addressed with an open industry data interchange standard.

## *Problem Statement*

Midstream pipeline operators are tasked with managing large volumes of constantly changing asset information. This data flows through the hands of numerous departments and service providers who all utilize different software applications and systems over the life of those assets. These disparate systems allow them to manage their workloads and make decisions relevant to their core mission based on best-available data. Those decisions can only be as good as their information is available, correct, and current.

Despite the frequency and complexity of information shared between parties, the pipeline industry lacks open standards for fully-interoperable data interchange (the ability for information to be exchanged across disparate systems without conversion or data loss). Considerable time, effort and money are spent converting and transforming this data to make it consumable by various data management systems. Additionally, this conversion process tends to introduce errors in the accuracy and loss to the density and resolution of data. The end result is time and money are wasted constantly transforming data instead of investing in the creation of open industry standards that allow data to flow between parties easier, faster, and cheaper.

Perhaps more importantly, these barriers to information sharing make it unnecessarily difficult for operators to maintain asset management systems that are Traceable, Verifiable and Complete (TVC). Following major pipeline industry incidents such as the one in San Bruno in 2010, regulatory agencies have mandated that operators maintain TVC-compliance in their asset data management practices and systems. The ability to track all decisions and activities that occur on all assets throughout their life from all involved parties requires their software and systems to freely and easily share data. Not only do all decisions and activities performed on all assets need to be shared, but the same applies to all decisions and activities performed on the data about those assets. They need to make all such data transparent, discoverable, and deliverable to the organization. This cannot be fully achieved without open, vendor-neutral data interchange standards.

## *PipelineML Definition*

The official definition of PipelineML is an initiative to develop an open extensible vendor-neutral international standard to enable the interoperable interchange of pipeline data between parties, disparate systems and software applications without loss of accuracy, density or data resolution; without ambiguity; and without need for conversion between intermediate or proprietary formats while maintaining contextual integrity. This standard is expected to leverage existing OGC standards such as GML, GML-SF, WFS, and WMS.

## *Get Involved*

There are several ways to get involved in the PipelineML data interchange standards development initiative. ***We welcome your active participation!***

- **Website:** First, visit our public-facing site at www.pipelineml.org. Sign up on the email news subscription list to ensure you stay informed of news and developments. Get involved on the site—read the blogs, documents, papers and engage in the online collaborative dialogue. Also, pass along information to colleagues who may be interested in this initiative.
- **Email:** Second, send an email to the PipelineML SWG Co-Chairs John Tisdale (jtisdale@eprod.com) and Jan Stuckens (jan.stuckens@merkator.be) indicating your interests, contact information and what resources you can bring to the initiative.
- **Interoperability Experiment:** Third, we will be conducting Interoperability Experiments (IE) as part of the OGC standards development process. If you wish to learn more about participating in an upcoming Interoperability Experiment, please contact Grace Cardenas (CGCardenas@eprod.com). You do not have to be an OGC member to participate.
- **OGC Membership:** Fourth, if you want to become involved in the detailed technical aspects of helping develop this standard, you must become a member of the Open Geospatial Consortium (according to the policies and procedures of OGC Standards Working Groups). You can find information on joining here. If you have questions about OGC membership, you may wish to contact some of the OGC staff who are familiar with the work of the PipelineML SWG — Trevor Taylor (ttaylor@opengeospatial.org) and Bart De Lathouwer (bdelathouwer@opengeospatial.org).

# Governing Design Principles

The following are the governing design principles established by the PipelineML Standards Working Group (SWG) in the initial months of its formation:

- **Practical** – A governing principle for the design and development of PipelineML is the need to solve practical business problems. Rather than pursue the perfect architectural design, we want to develop and deliver a solution that can be understood and implemented across the broad spectrum of stakeholders in the pipeline industry.
- **Flexible** – The standard shall be flexible enough to adapt to change. This flexibility is not achieved at the cost of sacrificing the other design principles. Instead, flexibility shall be designed within the context of all these governing design principles.
- **Fit-for-purpose** – Each element of PipelineML shall be focused on solving real-world business challenges facing those on the front lines of managing pipeline asset data. To fulfill this principle, a specific set of use cases shall be identified that are considered to be most fundamental to the pipeline industry. Our scope shall be focused on achieving these use cases and establishing fulfillment of those use cases with predefined test cases.
- **Interoperability** – Providing interoperability is essential to the success of this data interchange standard. A packet of PipelineML-compliant data shall remain consistent regardless of the data storage model or software application that produces or consumes it. Any PipelineML data packet can be consumed by other software application using any backend storage model without concern as to the platform or technologies used to produce it or consume it.
- **Unambiguous** – To achieve vender-neutral interoperability, any ambiguity must be removed from the data standard. If there is any room for interpretation of the meaning of any aspect of the interchange standard, variations may emerge in how vendors implement this standard. This would significantly dilute the value of this interoperable data interchange standard. As such, vigilance shall be applied to the process of removing ambiguity from the interchange standard.
- **Spatial Accuracy** – Spatial accuracy of the location of pipeline assets is essential to safe, efficient operations. Increasingly accurate GPS technologies are ubiquitous enough to facilitate a high level of resolution and spatial accuracy. The ability to accurately describe the location of assets via GPS is a critical governing design principle of this standards effort.
- **Vendor Neutral** – Standards are most effective when they do not favor any vendor solutions. By developing standards on a level playing field free of vendor favoritism, we create an ecosystem in which the needs of all stakeholders can be served. This fosters an environment of innovation. The

users and managers of data can freely pick and choose the right software solutions to suit their needs with the assurance they can interchange data in and out of those systems in a seamless and integrated manner because of these open standards. Such an ecosystem rewards innovative solutions to business needs.

- **Contextual Integrity** – This data interchange standard endeavors to provide enough granularity and specificity to provide contextual integrity. The standard shall ensure that when a PipelineML-compliant data packet is received by any party, the integrity of the original context of that data is preserved.
- **Extend & Embrace** – We do not want to reinvent the wheel in terms of defining existing data interchange standards unless existing standards do not exist or are inadequate to address the defined business needs. We shall utilize existing technologies such as UML, XML, XSD, GML, WMS, and WFS. By complying with these standards and extending them to service the needs of the pipeline industry, we are building on a solid mature foundation—thereby increasing our likelihood of success. We plan to utilize as many existing domain reference code lists as possible from such sources as ANSI, API, ASME, ASCE, etc.
- **Rapid ROI** – In as much as possible, PipelineML shall be designed to provide rapid return on investment for adopters. This is one of the key decisions to develop this data interchange standard under the umbrella of the OGC. By building PipelineML on top of the existing GML standards suite, we immediately gain the benefits of compatibility with all software applications that support these well-adopted standards.
- **Agile Lifecycles** – Although we will not follow a true agile software development methodology, we want to work toward small incremental deliverables that solve practical needs. We will start with a small scope that focuses on solving the most common data interchange needs. Once we solve the simplest and most urgent set of needs, then we will cycle through the process again with a larger scope that considers the next most urgent set of needs within the pipeline industry. With each cycle, we expect to capitalize on lessons learned from previous iterations.
- **Controlled Vocabularies** – Removing ambiguity also requires the use of common domain reference code lists (also known as controlled vocabularies). This ensures that all parties mean the same thing when they define features, types and attributes in PipelineML-compliant data packages. In as much as possible, existing code list standards shall be leveraged for this purpose. The American Petroleum Institute (API) is recognized as one of the leading standards bodies in defining reference code lists. Other standards shall be assessed and considered for adoption based on the needs of invested stakeholders. Additional discussion is needed regarding the translation of controlled vocabularies into other languages and units of measure.

The foundational technologies we have chosen to adopt are easy to understand and commonly accessible (in terms of the ability to acquire needed skills affordably). Some of the foundational interchange technologies we evaluated in 2012 were found to be too academic and involved obscure technology stacks. As such, they were not utilized to avoid an unnecessarily steep, expensive learning curve. Instead, we have selected the ideal technologies to meet these governing design principles.

## *Use Cases*

The current scope of the PipelineML standards development initiative is focused on fulfilling the following use cases:

### Use Case UC001: New Construction Survey

This release shall support the need to capture a detailed inventory of assembled pipeline componentry as defined in a survey conducted at the conclusion of new pipeline construction. This type of data package would be prepared by a survey and mapping company. This data packet would be provided to the pipeline operator. It would be imported into the operator's pipeline asset data management system. Supported componentry attributes shall be appropriate to the intrinsic properties knowable at the conclusion of pipeline construction.

### Use Case UC002: Pipeline Rehabilitation

This release shall support the need to capture a detailed inventory of assembled pipeline componentry as defined in a survey conducted at the conclusion of pipeline rehabilitation activities. This type of data package would be prepared by a survey and mapping company. This data packet would be provided to the pipeline operator. It would be imported into the operator's pipeline asset data management system. Supported componentry attributes shall be appropriate to the intrinsic properties knowable at the conclusion of pipeline rehabilitation activities. The Pipeline Rehabilitation use case is similar to the new construction use case, but includes items installed in typical rehabilitation activities such as repair sleeves, linepipe cut-outs and recoating jobs.

## *XML as a Foundation*

There are numerous loosely defined data interchange standards that are based on Microsoft Access, Excel, shapefiles or even CSV files. However, these solutions do not possess the capabilities to clearly define complex structures and rules while also facilitating flexibility, ease-of-use, openness, and extensibility. These and other requirements point to one platform that is well suited for this purpose. XML (eXtensible Markup Language) is the choice of most well-architected data interchange standards. XML provides a powerful suite of technologies that collectively enable us to describe pipeline assets and the activities performed on them in a clear flexible way. Since HTML, the language of the World Wide Web, is an extension of XML, there is a large global resource pool available who understand the fundamentals of XML. It is also human readable such that a person can open an XML file and understand the information being communicated.

One of the most powerful tools within the arsenal of XML is *XML Schema* (also known as XSD). This is a file written in XML that defines the structure and rules that must be followed for a package of data to conform to an interchange standard. Essentially PipelineML is a set of XML Schemas that define the structures and rules for describing pipeline assets and activities. For those familiar with relational databases such as SQL, an XML Schema defines the data structure in much the same way as a relational data model defines tables, columns, and datatypes. The key difference is that XML Schema provides much more powerful and flexible capabilities than those found in a relational data model (including object-oriented inheritance, extensibility, substitution groups, extensions/restrictions, inclusiveness/exclusiveness, collections, enumerations, dictionaries, controlled vocabularies, metadata, external links, etc.). The end user of PipelineML does not need to understand these concepts and technologies. Rather, the onus is on software vendors to add functionality such that their software can generate PipelineML (export) or consume PipelineML (import) asset data into their software based on these XML Schemas. The end user simply needs to select export to output a PipelineML file or import to consume one into their software application.

Upon its release and approval, we anticipate many software vendors building PipelineML import and export capabilities that follow the structures and rules of these XML Schemas. The end result will be one or more PipelineML XML files. These files can be shared between parties such as different departments within an operator. For example, a pipeline system designer may use AutoCAD software to design all the components needed to create a new pressurized pipeline system.

Once the project is complete and approved, they export all the detailed component information out of AutoCAD into a PipelineML file (in the event AutoCAD decides to support the PipelineML data interchange standard). The design department then forwards that PipelineML file to someone responsible for parts acquisition in preparation for construction.

In the next phase of the lifecycle, the parts acquisition group can import the supplied PipelineML file into their data management software application (provided its vendor supports PipelineML). Once imported, they have access to all the detailed information prescribed by the pipeline designer. This same process can continue through the entire lifecycle of the pipeline system including surveying, operations, integrity, regulatory reporting, divestiture, etc. Each time someone interacts with this set of data, they can add more data to it. For example, the person responsible for parts acquisition may add information from the manufacturer about pressure tests (i.e. MTR's) performed on each of those pipeline components. This information can accrue over time. By the time someone in operations receives the latest version of PipelineML file and imports it into their operational system, they have detailed information that has retained all collected historical knowledge accrued over time about those assets. This is the essence of TVC.

The scenario described above represents a mature state of the PipelineML data interchange standard. We will not support all this functionality in the initial release of the standard. In fact, we have strategically adopted a standards development methodology that starts small and grows as it matures. It will likely take years to achieve the scenario described above. Had we tried to support all possible industry use cases in the first release, the initiative would have bogged down under its own weight. Instead, we have chosen a proven strategy of beginning with a narrow scope of only the most critical use cases and growing it through short iterative lifecycles. Once we release the first version of the standard, we anticipate this milestone garnering attention throughout the oil and gas pipeline industry. This will help draw additional stakeholders to the initiative to provide necessary subject matter expertise in auxiliary spheres of interest to broaden and mature the standard over time. Throughout the development process, as stakeholders identify use cases outside current scope, we document them for potential inclusion in a future release of the standard. This iterative approach allows us to learn, adapt and build consensus over time.

## *Building on GML under OGC*

In 2013, we engaged in extensive dialogue with several international standards associations and bodies to assess best-fit for hosting this development effort (PODS, POSC Caesar, ASCE, ISO, OGC, etc.). After extensive dialogue throughout the year, we determined the Open Geospatial Consortium (OGC) was most compatible with our philosophies, needs, and goals. They follow a practical and effective approach to building consensus. They have a solid set of well-defined policies and procedures for standards development. Their support staff is strong and fully engaged. Their membership is comprised of global thought leaders from government agencies, universities, corporations, and technology firms. They have a proven track record of releasing data standards such as Geography Markup Language (GML, which was also adopted by ISO as 19136) with broad, diverse international adoption across many vertical markets. Also, GML contains many of the features we need to leverage such as a rich array of spatial data representation. GML is also built on XML and XML Schemas.

In March 2014, we attended the OGC Technical Committee meetings in Washington DC where we convened an ad-hoc meeting. Collectively, around 90 individuals attended the meeting in-person and remotely. By the end of the meeting, we received unanimous consent for the formation of the PipelineML Standards Working Group (SWG). Instead of building generic XML Schemas and XML files, we are now focused on building GML application schemas and GML files. A GML application schema is an XML schema that includes additional data structures and rules that must be followed. GML is extensible which means just as GML extends XML it can be extended via application schemas to meet specific industry needs. The end result of our work will be a set of PipelineML application schemas that are compliant with all GML and XML rules and extend both to include data structures and rules needed to describe pipeline assets and activities.

The biggest advantage of making PipelineML compliant with GML is once our standard is finished and approved by the OGC, it will work with any existing software application that supports GML (ESRI, QGIS, Grass, GeoServer, AutoCAD, Bentley, Intergraph, etc.). Since most smartphones in the world use GML-based standards (such as Web Map Services and Web Feature Services) for location services, this move automatically gives us broad device and application

adoption to at least visualize the location of pipeline assets. It should be noted that there are many flavors and versions of GML. Most software applications on the market only support the GML Simple Features Profile (GML-SF). The full GML standard contains some 600 pages of specifications. GML-SF contains less than 100-pages of specifications (thus making it far less onerous for software vendors to support).

Our goal is to bring PipelineML into compliance with GML-SF. There are three levels of Simple Features Profile compliance known as SF-0 (the simplest and most restrictive), SF-1 (slightly more complex and less restrictive) and SF-2 (the most complex and least restrictive). Our hope is to achieve GML SF-0 compliance since it is the broadest supported of all GML standards in the software market. However, it has yet-to-be determined whether we will be able to support all current and future use cases defined by pipeline industry stakeholders while adhering to this most restrictive level of compliance. We will not sacrifice needed use case support in order to achieve a certain level of GML-SF compliance. As such, the compliance level will be determined with each release of the standard. As we increase the number of use cases we support with each release, the compliance level we achieve may change from one release to another over time.

It should be noted that the PipelineML SWG is working with many other standards working groups within the OGC. Early in the initiative, with the help of OGC staff, we met with the leaders of many other DWG's and SWG's. We explained our needs, goals and philosophies. We evaluated whether the efforts of any other groups were close enough to our needs that we should consider merging with them. The Energy and Utilities Domain Working Group (DWG), the Land and Infrastructure DWG and SWG, CityGML, and 3 Dim were a few groups with some degree of overlap with our effort. After much discussion and consultation by OGC staff, we elected to proceed with our own independent initiative. That being said, we keep dialogue open with the chairs of other development groups within the OGC as well as a few others outside it. We are leaving the option open to eventually reach a point at which we could combine our efforts with those of another data interchange standards group. This shall be an issue under continual assessment. If we were to elect to integrate with another group, we shall endeavor to not lose any of the ground we have gained in supporting previously-defined industry use cases.

## *Schema Modularity*

One of the design philosophies of GML that is closely aligned with our approach is the use of schema modularity. Creating one PipelineML schema that contains all the data structures and rules needed for every possible pipeline data interchange use case would result in a massive file containing many hundreds of structures and rules. Such a large schema would take a considerable amount of time to process all those rules against a large dataset of pipeline assets each time someone wants to export their data into a PipelineML file. A better approach is to divide up those structures and rules into separate logically-modularized schemas.

Figure 1 – Package Dependencies or Modularized Schema Diagram in UML

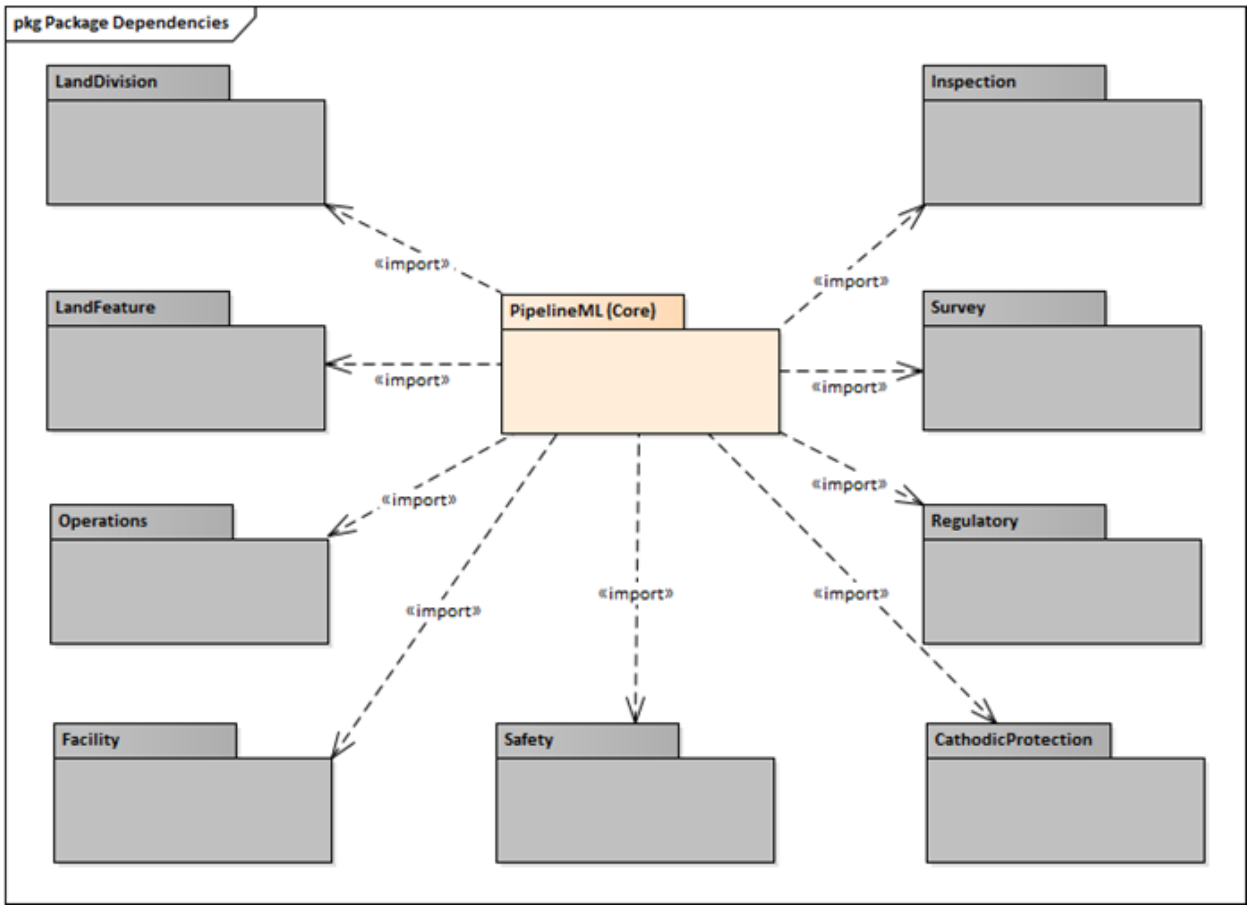



Figure 1 is a Unified Modeling Language (UML) diagram that illustrates this concept. It shows ten schemas as well as their dependencies we have defined so far (although they will likely grow in upcoming years). The center one is the Core schema that contains basic data structures and rules needed to define pipelines and the components from which they are built. It is focused on

defining physical objects and their intrinsic properties (that do not change over time). For example, manufacturers design components according to a set of specifications such as material type, nominal pipe size, outside diameter and wall thickness. These are unchanging intrinsic properties and thus would be described in the Core schema. Also, specific components may receive additional attributes following the manufacturing process such as a test pressure rating. Since these are intrinsic attributes, they would be described in the Core schema.

Transient component properties that change over time will be defined in other schemas, defined in the future. This will include attributes discovered through observations and measurements such as operating status, product/s being transported, product flow direction, operating pressure, regulatory classifications, corrosion levels and locations, whether a valve is in an open or closed state, etc. By providing clear, unambiguous differentiation as to what information is stored in each schema it is easier to understand which schemas are needed for any specific use case.

This approach to modularizing schemas allows a vendor who develops, for example, pipeline construction software to utilize this one Core schema to export pipeline construction data into PipelineML. Each of the other nine schemas is dependent upon the Core, imports it, and then extends or adds to it. A software vendor that builds Cathodic Protection data management systems not only needs the Core data structures and rules but also those found in the CathodicProtection schema. Since this schema automatically imports the Core schema, it combines both Core and CathodicProtection into a common set of data structures and rules. This approach simplifies the solution such that any software vendor only needs to reference and support those schemas that are applicable to the capabilities of their software.

Our goal with the first release of PipelineML is to only define the data structures and rules needed for the Core schema. The remaining schemas are colored dark grey to indicate they have been identified for future development but are not included in the initial release. Once the initial version of PipelineML is released and approved by the OGC, we will collectively prioritize the next highest set of use cases to include in a subsequent release of the standard. This approach allows us to gain momentum quickly and then gradually broaden the scope of PipelineML to eventually include all possible data interchange use cases needed by the international oil and gas pipeline industry. The end result will be a set of PipelineML schemas that are all designed to work together (without redundancy or conflicting structures and rules) as a cohesive collective. Any of these individual schemas can be updated in future releases of the standard as discoveries are made by stakeholders.

# Key Technical Elements

## *Overview of Conceptual Model*

Most of our work in the first two years of the OGC PipelineML SWG was spent working through numerous rapid lifecycles of encoding actual pipeline asset data into ever maturing prototypes of PipelineML. After two years of prototyping, we had a strong handle on the essential business needs and technical issues. Next, we started back at the beginning by defining a conceptual model based on all the lessons learned. Figure 2 is a UML class diagram of our current conceptual model for the Core schema. It should be noted that this is not a final version and as such no development work should be undertaken on it. This overview is intended to inform new project stakeholders on the current state of our standards development initiative. Although the underlying class model contains detailed attribution and datatypes, this overview will only cover the highest level objects. Once this model is finalized, we will generate the PipelineML Core schema from it (and make manual tweaks as needed).

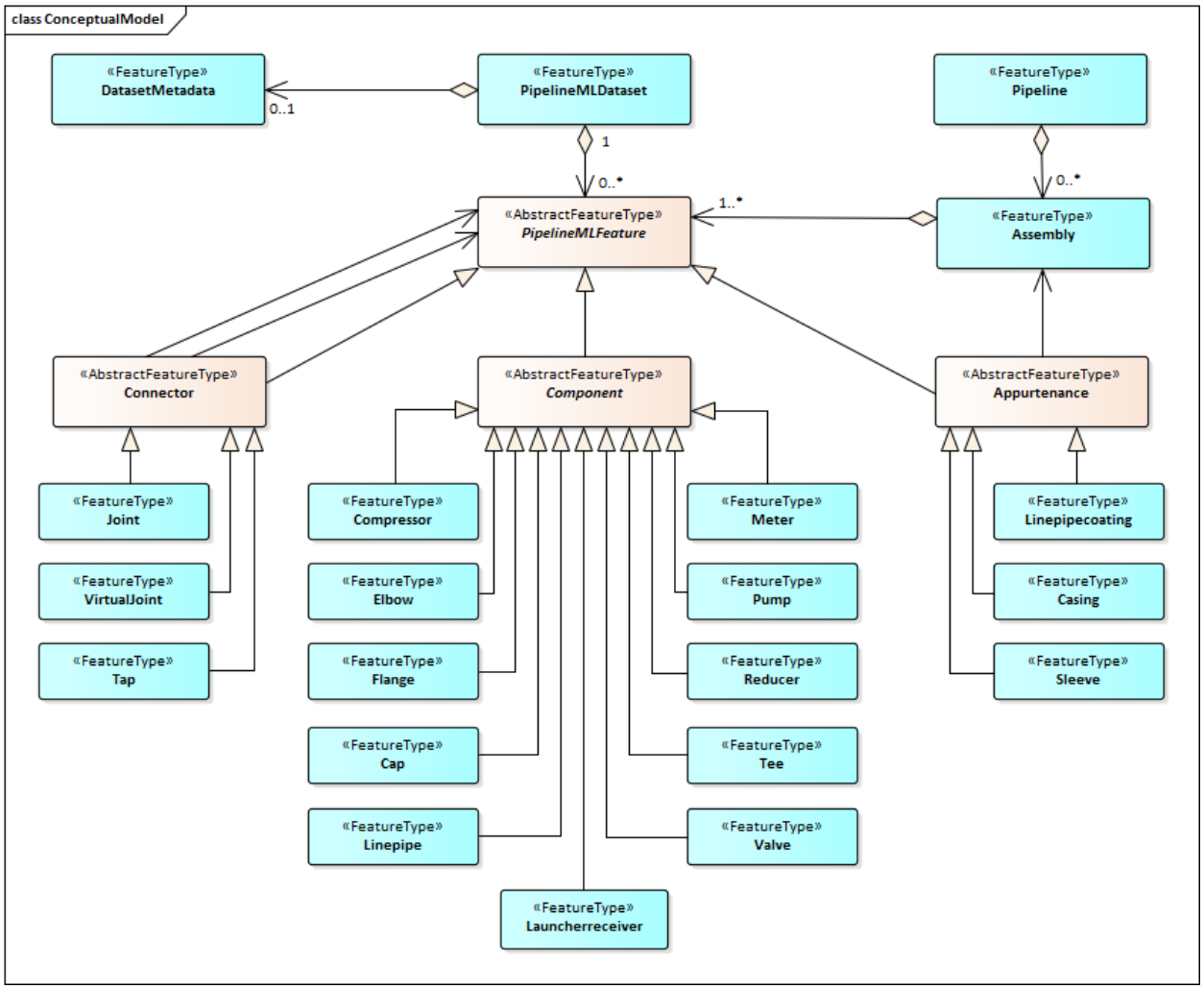
The light blue boxes (along with the type definition of FeatureType) indicate classes that get instantiated into element tags in PipelineML. The light beige boxes (along with the type definition of AbstractFeatureType) indicate abstract classes that serve a different purpose and do not get translated into XML element tags. Starting with the top center box, we have the root element—PipelineMLDataset. Since XML requires there to be one and only one root element, we use this generic placeholder to service this purpose. If we were to use another element such as Pipeline, this would automatically restrict us to supporting only a single pipeline of data stored in a PipelineML file. Since we want to allow as much flexibility as possible and not inject any unnecessary artificial restrictions on the design, we utilize this generic root placeholder.

The light blue box entitled DatasetMetadata is a FeatureType class that can be instantiated into an XML element tag. The beige diamond on the line that connects it to the PipelineMLDataset object indicates this to be an aggregation relationship. This means that DatasetMetadata is aggregated into PipelineMLDataset. The text (0..1) on the left indicates that there can be either 0 or 1 metadata element contained within the PipelineMLDataset. This means the inclusion of metadata is optional (as there are specialized use cases where metadata in the Core schema may not be necessary). However, we consider it a best practice to include metadata information within the

PipelineML dataset in most use cases. If metadata is provided, there can be only a single set of metadata information. The details of what data may be included in the metadata section are covered in the technical specification documentation currently under development.

Below the PipelineMLDataset class is a PipelineMLFeature class. This is beige in color and of the class type AbstractFeatureType which means it is not actually instantiated into an XML element tag. This means we will use this abstract class to help define the structure of the data but it will not actually appear in a PipelineML file (more on this issue later). The beige diamond on the line connecting these two classes indicates this to be an aggregation relationship. The text next to this object (0..*) indicates there can be 0 to an infinite number of PipelineMLFeature objects associated with a PipelineMLDataset.

**Figure 2 – PipelineML Conceptual Model in UML**



Since DatasetMetadata and PipelineMLFeature are the only two class objects connected to PipelineMLDataset, we could say that PipelineMLDataset is an aggregation of up to one DatasetMetadata element and up to an infinite number of PipelineMLFeature objects. Since both DatasetMetadata and PipelineMLFeature class objects are optional (0..), this means that a PipelineML file could, in theory, contain nothing more than the PipelineMLDataset tags. This may seem nonsensical but it is a strategic part of the design. Since this is the Core schema and other auxiliary schemas will import the Core, it is possible that some PipelineML data files will not need to include any component data at all. For example, if a file is created to define a pipeline's Right of Way location, that file will not need to include any component-level details. This means ancillary schemas will not be negatively impacted in terms of flexibility or scalability by integrating with the PipelineML Core schema. By making all features in the Core schema optional, it provides a great deal of flexibility in how additional PipelineML schemas are designed and implemented in the future. The advantages of this approach will become clear over time.

To the right is the Assembly class. The beige diamond next to the Assembly object indicates an aggregation of 2..* (between 2 and an infinite number) of PipelineMLFeature objects. Since it is of the type FeatureType, it can be instantiated into an element tag in PipelineML. We developed the concept of an Assembly to be 2 or more components but less than an entire pipeline system of components that may share some common attributes and thus warrant the inclusion into an Assembly. An Assembly is one of the only abstract (non-physical) elements in the Core schema. This could be a mainline assembly, an assessment segment, a collection of components involved in a rehabilitation project, etc. Whether the list of Assembly types will be loosely or well defined and controlled has yet to be determined.

The class above is that of a Pipeline. It is of the type FeatureType and thus can be instantiated into an element tag in a PipelineML file. This would usually reference a pipeline system (although exactly how a Pipeline is defined is open to interpretation). We will not apply any artificial constraints on this class that enforces any specific interpretation of a Pipeline. You can see that the relationship between an Assembly and a Pipeline class is that of an aggregation. Thus, a Pipeline is an aggregation of (0..*) zero to an infinite number of assemblies. Therefore, it is possible to define a Pipeline in PipelineML that contains no assemblies. Here too, we designed as much flexibility into the model as possible so as to support a broad set of needs.

If we move back to the PipelineMLFeature abstract class, you see 3 additional abstract classes pointing to it with filled arrows on the PipelineMLFeature end. This indicates a generalization relationship. This means that PipelineMLFeature is the superclass or parent and that the Connector, Component and Appurtenance are subclasses or children that inherit the properties (and potentially the methods) of PipelineMLFeature. What this class structure does for us is allows us to define a set of common traits or attributes that all types of components share in common. For example, it is likely that all pipeline components commonly share the following attributes: ID, Code, Description, Location, InstallDate, and RemoveDate. If we define these 6 attributes in the PipelineMLFeature abstract super class, then everything that gets inherited from it automatically acquires these attributes. This means that instead of having to define these 6 attributes for each and every type of component in PipelineML, we simply define them once in this superclass and all inheriting components automatically acquire these attributes. This results in a much smaller schema (that therefore processes data imports and exports faster) that is also more readable and manageable.

Below is the Connector abstract class. This means it will not be directly instantiated into an element tag in a PipelineML file. It automatically inherits all the attributes of its parent PipelineMLFeature. As already mentioned, there is a connecting line with the filled arrow that indicates a generalization or inheritance relationship. Additionally, we also have two adjacent lines with non-filled arrows. These indicate association relationships. This is because the role of a Connector is to connect two components together. So, the two lines pointing back to PipelineMLFeature indicate this Connector class can define a connection between any two types of components that inherit from the PipelineMLFeature class. The reason we have the Connector abstract class is to give us another opportunity to define some common data structures all classes that inherit from it will acquire. Let's say that all types of connecting components share the following attributes in common: ConnectionType, StartComponentID, and EndComponentID. If we assign these attributes to the Connector abstract class, all child classes that inherit from it will gain these attributes. Once again, this approach provides for greater scalability, readability and manageability.

Below the Connector abstract class is the Joint class. Since it has a generalization relationship, it inherits all the attributes of its parent Connector as well as those of the PipelineMLFeature super class. As a result, before we begin defining any attributes that are specific to a Joint, the Joint class has already inherited the following attributes: ID, Code, Description, Location, InstallDate, RemoveDate, ConnectionType, StartComponentID, and EndComponentID. A component Joint

might also include a unique attribute called JointType that could contain such values as Welded, Bolted, Threaded Coupling, etc. A Joint component might also be assigned the unique attribute of JointCoatingType.

It should be noted that many use cases do not require the use of connectors or joints. Older vintage pipeline systems may have no information on the types of joints used to connect components. In such cases, there are no requirements that connections or joints be defined in PipelineML data packages. They are entirely optional. As IMU-enabled inline inspections call out joints and welds, whatever information is determined about them can be communicated in PipelineML. In the case of new construction in which detailed information is known about the connections and joints, we can define as much detail as needed. By making virtually all attributes optional, we make the standard easy to implement and support a wide variety of needs (use cases).

Whether component connections are used or not, the vertices that define the beginning and ending of each component (when 2 dimensional linestrings are used) provide enough information to derived connectivity. That is, if the coordinates for the end of one component coincide with those of the end of an adjacent one (including the potential use of a buffer), we can imply connectivity—thus allowing us to dynamically build network topology. Since pipeline systems are primarily linear, we believe the best practice is to utilize this form of dynamic connectivity and network topology. In other standards such as CityGML, non-linear structures like buildings introduce more complex dynamics. For example, without a logical connectivity model it is not possible to know that 10 floors of an office building are all connected via an elevator based solely upon spatial node coincidence.

It is possible to define logical connectivity within this PipelineML conceptual model to explicitly articulate whether one component is connected to another. History has demonstrated the difficulty in attempting to manage both coincidental connectivity and logical connectivity. Over time, they tend to drift apart. When there is a delta or conflict between the two models, decisions have to be made as to which is correct and undertake the work of bringing the two systems back into alignment. Given this trend, our recommended best practice is to utilize coincidental connectivity as the primary determination of network topology.

There are special situations in which coincidental connectivity alone is inadequate to address the needs of pipeline configurations. This is particularly true of facilities (processing plants, tank racks, pump stations, etc.). In most cases, asset data management systems do not store all the componentry located inside the fence of a facility. All that may be known is that a particular facility is connected to a particular pipeline system and approximately where. Exactly which two components are connected is not a use case that affects most business units. This presents a good use case for augmenting a coincidental connectivity model with a logical one. This is why we introduced the VirtualJoint class. This class can be instantiated into an element tag that explicitly defines connectivity (such as a pipeline connecting to a facility somewhere within the fence) so as to facilitate needed business decision support. This class may contain some unique attributes such as a VirtualJointType and perhaps a specialized geometry field that indicates approximately where this connection occurs (the details are still being worked out).

The Tap class is similar to Joint in that it inherits the properties of the Connector class. Yet, it is used to define where a linepipe was tapped to attach instrumentation, redirect the flow of product, etc. This class might include additional attribution pertinent to a tap such as those shown in the right column of Figure 3. The result of this design model makes all 3 columns of attributes available to describe the intrinsic properties of a particular Tap (these lists of attributes are still under review and are likely to change prior to the publication of this data interchange standard). We make all these attributes optional so that only those required by XML and GML must be provided.

**Figure 3 – Sample Attribution of Tap Component**

| Attributes Inherited from PipelineMLFeature | Attributes Inherited from Connector | Attributes Uniquely Assigned to Tap |
|---|---|---|
| ID | ConnectionType | TapManufacturer |
| Code | StartComponentID | TapManufactureDate |
| Description | EndComponentID | TapDimensions |
| Location | | TapInletInsideDiameter |
| InstallDate | | TapOutletOutsideDiameter |
| RemoveDate | | TapFunction |
| | | TapInstallMethod |
| | | TapNominalPipeSize |

The Component abstract class is similar to the Connector class in that it inherits from the PipelineMLFeature super class. The primary trait all members of the Component class have in common is that they are coterminous in nature. That is, they physically connect together in order to collectively form a pressurized pipeline system through which product may flow. All attributes that are commonly shared among coterminous components can be assigned to this class and therefore inherited by all child classes. This might include the attributes shown in the middle column of Figure 4. Keep in mind that these attributes lists have not been finalized and changes are likely.

**Figure 4 – Sample Attribution of Linepipe Component**

| Attributes Inherited from PipelineMLFeature | Attributes Inherited from Component | Attributes Uniquely Assigned to Linepipe |
|---|---|---|
| ID | Manufacturer | LinepipeTestPressure |
| Code | ManufactureDate | LinepipeSecondaryWallThickness |
| Description | | LinepipeAncestorID |
| Location | | LinepipeBendType |
| InstallDate | | LinepipeDimensions |
| RemoveDate | | LinepipeOutsideDiameter |
| | | LinepipeNominalPipeSize |
| | | LinepipeWallThickness |
| | | LinepipeSeamWeldOrientationClockPosition |
| | | LinepipeSeamWeldOrientationAngle |
| | | LinepipeSeamWeldType |
| | | LinepipeCharacteristics |
| | | LinepipeSpecification |
| | | LinepipeGrade |
| | | LinepipeMaterial |
| | | LinepipeYieldStrength |

Each of the coterminous components inherits from the Component class and extends it to include additional attributes specific to each component type: Compressor, Elbow, Flange, Cap, Linepipe, Launcherreceiver, Meter, Pump, Reducer, Tee, and Valve. What makes Linepipe especially flexible is that it can be used to define a specific piece of linepipe (i.e. 80 foot) or it can be used to define an entire pipeline system where the division and location of individual components is not known. This mechanism allows for the generalization of componentry, as needed. It creates an easy way for an operator who is working with a storage model that does not support component-

granularity to utilize PipelineML and gradually grow into its more robust capabilities as they increase the resolution of their asset data.

The final section of classes in this UML model is Appurtenance. This class represents components that are placed on coterminous components or could not exist without the underlying coterminous components. Appurtenances may share such common attributes as StartComponentID and EndComponentID. These attributes are inherited by the child classes Linepipecoating, Casing, and Sleeve.

Note that the Appurtenance class has an association relationship with the Assembly class. The thought behind this design concept is that a group of components may be defined as an Assembly and then an appurtenance such as a Linepipecoating may be associated with that Assembly to indicate the group of components it covers. An alternate approach is to simply define the StartComponentID and the EndComponentID that are covered by the appurtenance (thus implying all components between the start and end participate in the appurtenance).

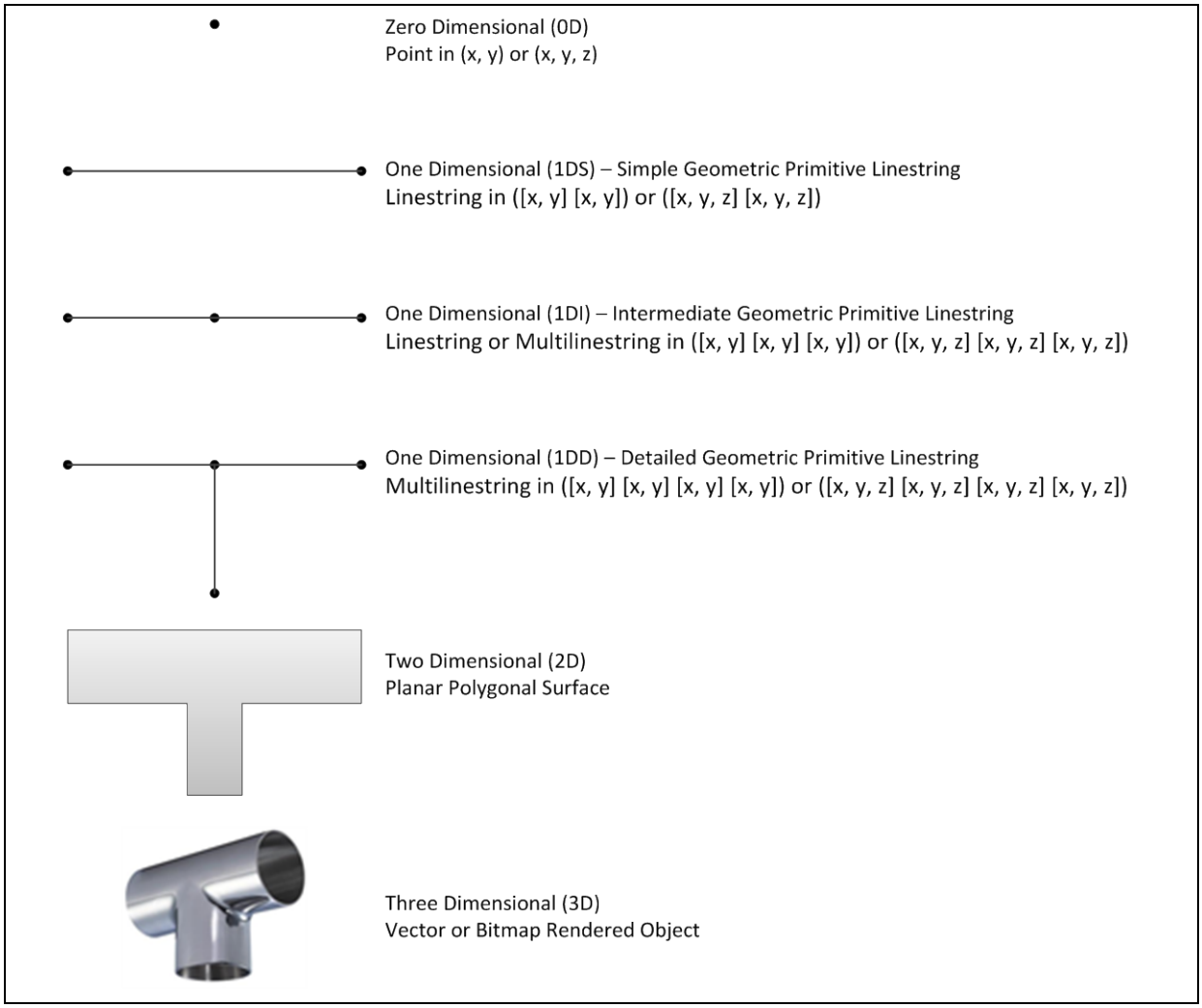
## *Component Geometry Representation*

Another consideration in the architecture of PipelineML is how best to represent the geometry of individual components. For example, some pipeline data management systems store the geometry of valves as points whereas others do so as linestrings. Both are legitimate use cases and should be supported by this data interchange standard. After detailed discussions with many of the other OGC standards working groups, we identified a best practice methodology. Instead of limiting component geometric representation to a single solution, we will use substitution groups in XML to support a robust assortment of alternatives. In fact, the same component could be represented in two or more of these geometry types in a single file, based on the capabilities of the software that exports it.

Figure 5 illustrates options for geometric representation of a Tee. The simplest approach is to represent components as points (using 2 or 3 coordinate points depending upon whether the elevation coordinate is supported by a given software application) based on the centroid of the component at center of pipe. We have not worked through how many variations of linestring will be supported. Some of these variants are linestrings while others are multi-linestrings. The most favored is the use of the 1DD variant. However, we may decide to support multiple geometric primitive

variations of one dimensional linestrings. The use of 2-dimensions that utilize a polygon based on a planar view of the component will also be supported. A software application that visually renders data from a PipelineML file would likely display 0 or 1-dimensional representation for low zoom levels. Once the user zooms to an appropriate level, the display could switch to rendering geometry from the 2-dimensional representation. We will not support 3-dimensional geometric representation of components in the first release of PipelineML (in terms of 3D vector or bitmap shaded rendering, although we will support the use of 3-dimensional representation in terms of x, y, z coordinates—and potentially *m* as the fourth—although discussion around that topic continues). However, we expect to include this capability in a future release.

**Figure 5 – Geometric Representations of a Tee Component**



## *Pipelines versus Componentry*

Another design decision was whether to support the definition of pipelines as generic entities (i.e. a single 500-mile pressurized entity) or as a set of individual components (linepipes, flanges, elbows, meters, valves, etc.) that collectively comprise a 500-mile pressurized system. We concluded the standard must support both scenarios. Most pipeline operators have a broad vintage of assets. Pipelines many decades old define less granular information on the individual components utilized in their construction. Newer systems contain detailed component information. As technologies used in inline inspection tools continue to mature, they provide robust means to bridge this information gap. Over time, pipelines with limited component information will become increasingly denser in resolution. Another factor in this trend is how regulatory agencies continue to drive the industry toward reporting higher-resolution data. PipelineML is designed to address all these possible levels of data granularity and continue growing in the future alongside the technologies it utilizes.

## *Linear Reference versus Global Positioning*

Another important decision discussed extensively is whether to support a Linear Reference System (LRS), Global Positioning System (GPS), or both. One of the challenges with using a LRS is the lack of standards. Numerous incompatible LRS systems and homegrown variants are in use. GPS is highly standardized and there are common conversion routines available for the various projections. This makes GPS an attractive option. Instead of trying to design a data interchange standard that attempts to accommodate every possible LRS, a simpler approach is to have software that supports linear referencing convert location data into common GPS coordinates. Then, when PipelineML data is imported back into a system that utilizes linear reference, it can convert the GPS coordinate data into its particular flavor of LRS.

## *Domain Reference Code Lists*

For an interchange standard to achieve interoperability, it must provide a mechanism to ensure all parties use common terminologies to communicate the same concepts. Without this safeguard, two parties could be communicating different concepts using the same language or using different language to communicate the same concepts (thus requiring human intervention to recode or

translate terminologies as part of the interchange process). A well-designed data interchange standard should remove ambiguity so as to facilitate machine-to-machine communication without the need for human manipulation. This issue is commonly addressed through the use of domain reference code lists or technically known as *controlled vocabularies*. Controlled vocabularies restrict values that can be provided for any given data placeholder based on a list of explicit values. There are more advanced XML tools available such a Schematron that could be utilized to define complex rules instead of lists. However, this is beyond the scope of the first release of this initiative.

XML provides many solutions to support controlled vocabularies. The first method uses enumeration to hardcode a list of acceptable values for each data field directly into the schema. The problem with this approach is that once a schema has been approved by a standards body such as the OGC, it must go through an arduous process to approve any changes. Each time a change is made to one of the embedded code lists or values, even a single change would require the schema to be reapproved (which can take months). This makes enumeration impractical for this purpose. The better approach is to manage vocabularies outside the schema. Each data field in the schema may point to the location (URI) of an externally controlled vocabulary. When a software application is exporting or importing PipelineML data, it validates data values against these external vocabularies.

We will manage these external controlled vocabularies as static XML files that conform to an XML vocabulary standard as a short-term solution. They will be published under a naming authority such that the URI for each controlled vocabulary XML file will always remain consistent. Eventually, we plan to have a web-based type registry created that will allow these vocabularies to be managed in an effective and collaborative manner (using authentication, user roles, notification system, approval workflow, historical change tracking, etc.). During data export or import, the software application would check relevant data values against the type registry as part of the validation process. If exceptions are found, the user would be notified and given the opportunity to correct them. Invalid values may be allowed but flagged as substandard (such issues are still under review and discussion).

In terms of who will be responsible for the management of these controlled vocabularies, the OGC PipelineML Standards Working Group will do so in the short-term. We will eventually select an existing entity comprised of subject matter experts or help form one that will be responsible for the ongoing management of these vocabularies.

## *Bound versus Unbound Attribute Values*

Another design consideration of PipelineML was whether to maintain each list of reference codes as separate, isolated values (unbound) or to bind the values together between these lists (bound) based on valid data combinations. For example, we could assign the following 4 attributes and values to a linepipe component:

- Linepipe Specification = API-5L
- Linepipe Grade = X-70
- Linepipe Material = Low Carbon Steel
- Linepipe Yield Strength = 70,000

A more efficient approach is to define the acceptable combinations of attributes and values. Consider that X-70 linepipe grade with yield strength of 25,000 is not valid and such linepipe has never actually been manufactured. So, although each of these values could be individually assigned to these two attributes, ideally they should not be allowed. Rather than create complex rules around which value combinations are allowed, the simpler approach is to create a bound code type that contains acceptable attribute/value combinations to interrelated attributes. Then, a single code could be used to identify this combination of attributes and values. For example, suppose the following attributes and values were the only valid combinations of linepipe specification, grade, material, and yield strength that were ever manufactured:

| Code | Specification | Grade | Material | Yield Strength |
|---|---|---|---|---|
| 3 | API-5L | A-25 | Low Carbon Steel | 25,000 |
| 6 | API-5L | B | Low Carbon Steel | 35,000 |
| 9 | API-5L | X-46 | Low Carbon Steel | 45,000 |
| 10 | API-5L | X-46 | Low Carbon Steel | 46,000 |
| 11 | API-5L | X-52 | Low Carbon Steel | 50,000 |
| 12 | API-5L | X-52 | Low Carbon Steel | 52,000 |
| 13 | API-5L | X-56 | Low Carbon Steel | 56,000 |

| | | | | |
|---|---|---|---|---|
| 14 | API-5L | X-60 | Low Carbon Steel | 46,000 |
| 15 | API-5L | X-60 | Low Carbon Steel | 60,000 |
| 16 | API-5L | X-65 | Low Carbon Steel | 65,000 |
| 17 | API-5L | X-70 | Low Carbon Steel | 70,000 |

Instead of assigning these values independently as LinepipeSpecification = API-5L, LinepipeGrade = X-70, LinepipeMaterial = Low Carbon Steel, and LinepipeYieldStrength = 70,000, we could simply assign the value LinepipeSpecificationBoundCode = 17. This is not only a more concise way to represent the same information, but also creates a mechanism for enforcing complex business rules in a simple manner.

There are drawbacks to only supporting the use of bound attribute values. First, most operational systems do not store data in this manner. Also, these lists would require management overhead. Furthermore, a sizable amount of data currently stored in asset management systems contains erroneous or substandard data (including many unknown values or invalid combinations that have not yet been discovered and corrected). This would require a clean-up effort before operators could begin utilizing PipelineML (which would obviously create a barrier-to-adoption).

Given these factors, we believe the ideal approach is to provide mechanisms to support bound and unbound attribution values. Bound values would be considered standardized, best practice, and authoritative data. Unbound values are fully supported but PipelineML-enabled software applications may wish to flag such unbound data as substandard (requiring further vetting or engaging more complex logic to resolve conflicts in the attributes and values). This ability to support both bound and unbound data also provides an opportunity for software vendors to develop applications that upon consuming PipelineML, can establish business rules to govern the processing of bound and unbound data. For example, it may allow bound data to be automatically accepted through the first gate of data staging. Unbound data could be flagged as requiring additional business processes for its acceptance.

# Appendix

## *Examples of PipelineML*

Let's look at some examples of PipelineML to help demystify it and see how practically it addresses the need to share pipeline asset information. Since we have not finalized PipelineML, these are rough examples and any of the syntax used is subject to change prior to approval of this data interchange standard. Some of the less meaningful syntax is removed to simplify and help illustrate the principle concepts.

### Example 1 – A single piece of linepipe in PipelineML

```
<PipelinMLDataset>
  <Linepipe gml:id="ID4E5250A1-72FD-E411-80C9-38EAA735D691">
    <location1D>
      <gml:lineString gml:id="ID4E5250A1-72FD-E411-80C9-38EAA735D691Loc1D">
        <gml:posList>29.766912361000038 -95.350627409999959 29.766918221000026 -
95.35062981699997</gml:posList>
      </gml:lineString>
    </location1D>
  </Linepipe>
</ PipelinMLDataset >
```

In example 1, we see the default root element PipelineMLDataset. Next, there is no metadata, no pipeline and no assembly data provided (as we are starting with a minimalistic example). We have a single component listed that is a Linepipe. All features in GML must have an ID property that is unique in the entire file (since this ID field is part of the GML specification, it is listed under the gml namespace, thus gml:id). So far, we are standardizing ID values as database GUID's. However, since the GML schema defines ID as a string, it cannot begin with a number and since database GUIDs sometimes begin with a number, we prefix it with the characters *ID* (discussion on the ID character prefix is ongoing—alternative considerations are to use an abbreviation for the component type, LP in this case, or perhaps a unique string assigned to each company/operator, this latter approach closely aligns with the design of the European INSPIRE standard).

GML defines a capitalization convention that feature elements begin with a capitalized letter and attributes begin in lower case. Next, we have a location element that provides the geospatial

coordinates where this component is located as a 1-dimensional linestring. Since GML also requires the geometry to contain a gml:id, we are suffixing the same component ID with *Loc1D* for 1 dimensional location such that numerous locations can be defined for a single component/feature. Everything inside the Location1D elements is conventional GML location information. It provides the location as a linestring via x y coordinates as [x, y] [x, y], however it could have included a third elevation coordinate thus supporting 3 coordinate dimensions as ([x, y, z] [x, y, z]). The remaining elements are simply the closing of hierarchical XML-based brackets.

### Example 2 – A single linepipe record with full sample attribution in PipelineML

```
<PipelinMLDataset>
  <DatasetMetadata>
    <packageVersion>1</packageVersion>
    <packageName>PipelineML Sample Data Package 020</packageName>
    <packageDescription>This package is for testing early prototypes of exchanging pipeline construction data via PipelineML.</packageDescription>
    <packageNote>This information is for testing purposes only and should not be used in production data management systems. Its origin was as public railroad data converted into a non-proprietary pipeline segment for public distribution. It is intended to accurately represent pipeline data. Detailed attribution is in progress.</packageNote>
    <interchangeStandard>PipelineML</interchangeStandard>
    <interchangeStandardVersion>0.0.1</interchangeStandardVersion>
    <beginCreationDateTime>2016-06-28T09:32:44.947</beginCreationDateTime>
    <endCreationDateTime>2016-06-28T09:32:44.951</endCreationDateTime>
    <languageID>0</languageID>
    <languageCode>us_english</languageCode>
    <unitOfMeasure>U.S. Feet</unitOfMeasure>
    <company1Name>Morris P. Hebert, Inc.</company1Name>
    <company1Role>Data Survey Source</company1Role>
    <company1Department>Survey and Mapping</company1Department>
    <company1Contact>Terry Strahan</company1Contact>
    <company1ContactTelephone>713-219-4419</company1ContactTelephone>
    <company1ContactEmail>tstrahan@mphinc.com</company1ContactEmail>
    <company1ContactAddress>10101 Southwest Fwy, St 620, Houston, TX 77074</company1ContactAddress>
    <company2Name>Moore Resource Systems</company2Name>
    <company2Role>Data Processing Source</company2Role>
    <company2Department>Data Services</company2Department>
    <company2Contact>Rod Burden</company2Contact>
    <company2ContactTelephone>519-722-0697</company2ContactTelephone>
    <company2ContactEmail>rod.burden@enghouse.com</company2ContactEmail>
    <company2ContactAddress />
```

```
    <company3Name>Enterprise Products</company3Name>
    <company3Role>Data Processing Source</company3Role>
    <company3Department>Asset Data Management</company3Department>
    <company3Contact>John Tisdale</company3Contact>
    <company3ContactTelephone>713-381-5565</company3ContactTelephone>
    <company3ContactEmail>jtisdale@eprod.com</company3ContactEmail>
    <company3ContactAddress>1100 Louisiana St, Houston, TX 77002</company3ContactAddress>
    <pipelineID>9AF16960-1A90-DF11-9C20-0026552185C4</pipelineID>
    <pipelineName>Sample Creek to Test City</pipelineName>
  </DatasetMetadata>
  <LinePipe gml:id="LP4E5250A1-72FD-E411-80C9-38EAA735D691">
    <gml:name>LinePipe 9637</gml:name>
    <componentID>4E5250A1-72FD-E411-80C9-38EAA735D691</componentID>
    <componentCode>9637</componentCode>
    <linePipeSpecificationBoundCode>17</linePipeSpecificationBoundCode>
    <linePipeSpecificationBoundName>API-5L X-70 Low Carbon Steel
70000</linePipeSpecificationBoundName>
    <linePipeSpecificationSpecification>API-5L</linePipeSpecificationSpecification>
    <linePipeSpecificationGrade>X-70</linePipeSpecificationGrade>
    <linePipeSpecificationMaterial>Low Carbon Steel</linePipeSpecificationMaterial>
    <linePipeSpecificationYieldStrength uom="pound">70000</linePipeSpecificationYieldStrength>
    <linePipeDimensionBoundCode>157</linePipeDimensionBoundCode>
    <linePipeDimensionBoundName>10.750 0.307 10.000</linePipeDimensionBoundName>
    <linePipeDimensionOutsideDiameter uom="inch">10.750</linePipeDimensionOutsideDiameter>
    <linePipeDimensionNominalWallThickness uom="inch">0.307</linePipeDimensionNominalWallThickness>
    <linePipeDimensionNominalPipeSize uom="inch">10.000</linePipeDimensionNominalPipeSize>
    <linePipeBendTypeCode>5</linePipeBendTypeCode>
    <linePipeBendTypeName>Straight Pipe Joint</linePipeBendTypeName>
    <linePipeSeamWeldTypeCode>2</linePipeSeamWeldTypeCode>
    <linePipeSeamWeldTypeName>Electric Resistance Weld</linePipeSeamWeldTypeName>
    <linePipeMillTestPressure uom="pound per square Inch"></linePipeMillTestPressure>
    <linePipeSecondaryWallThickness uom="inch"></linePipeSecondaryWallThickness>
    <description>Linepipe component 10” API-5L X-70 Low Carbon Steel 70000</description>
    <comment>This is a sample linepipe component with full sample attribution. This attribution data is from an
earlier prototype and may not accurately represent the latest parameters.</comment>
    <installDate>2003-08-04</installDate>
    <removeDate>2015-06-18</removeDate>
    <manufactureDate>2002-02-26</manufactureDate>
    <manufacturerCode>95</manufacturerCode>
    <manufacturerName>Stupp Steel</manufacturerName>
    <location0D>
      <gml:point gml:id="LP4E5250A1-72FD-E411-80C9-38EAA735D691Loc0D" srsName="EPSG:4326">
        <gml:pos>29.766915291000036 -95.350628613499936</gml:pos>
```

```
        </gml:point>
      </location0D>
      < location1D>
        <gml:lineString gml:id="LP4E5250A1-72FD-E411-80C9-38EAA735D691Loc1D"
srsName="EPSG:4326">
          <gml:posList>29.766912361000038 -95.350627409999959 29.766918221000026 -
95.35062981699997</gml:posList>
        </gml:lineString>
      </ location1D>
  </LinePipe>
</ PipelinMLDataset >
```

In this example, an assortment of metadata information has been added to demonstrate some of the concepts through which we are working. Next, we have a linepipe component that contains both a 0-dimensional point representation as well as a 1-dimensional linestring. A complete listing of potential linepipe attributes are listed (all under extensive review by the PipelineML SWG).

### Example 3 – Three component records with geometry attribution in PipelineML

```
<PipelinMLDataset>
  <LinePipe gml:id="LP9D5A50A1-72FD-E411-80C9-38EAA735D691">
   <gml:name>LinePipe 11764</gml:name>
   <location0D>
    <gml:Point gml:id="LP9D5A50A1-72FD-E411-80C9-38EAA735D691Loc0D " srsName="EPSG:4326">
     <gml:pos>30.094681046501208 -96.079921032</gml:pos>
    </gml:Point>
   </ location0D>
   < location1D>
    <gml:LineString gml:id="LP9D5A50A1-72FD-E411-80C9-38EAA735D691Loc1D" srsName="EPSG:4326">
     <gml:posList>30.094680668000024 -96.079938714999969 30.094681425000033 -
96.079903348999949</gml:posList>
    </gml:LineString>
   </ location1D>
  </LinePipe>
  <Joint gml:id="JNTC65650A1-72FD-E411-80C9-38EAA735D691">
   <gml:name>Joint 10781</gml:name>
   < location0D>
    <gml:Point gml:id="JNTC65650A1-72FD-E411-80C9-38EAA735D691Loc0D" srsName="EPSG:4326">
     <gml:pos>30.09536126200004 -96.079954878999956</gml:pos>
    </gml:Point>
   </ location0D>
  </Joint>
```

```
  <LinePipe gml:id="LPC65650A1-72FD-E411-80C9-38EAA735D691">
    <gml:name>LinePipe 10781</gml:name>
    < location0D>
      <gml:Point gml:id="LPC65650A1-72FD-E411-80C9-38EAA735D691Loc0D" srsName="EPSG:4326">
        <gml:pos>30.095294677000172 -96.0799493344962</gml:pos>
      </gml:Point>
    </ location0D>
    < location1D>
      <gml:LineString gml:id="LPC65650A1-72FD-E411-80C9-38EAA735D691Loc1D" srsName="EPSG:4326">
        <gml:posList>30.09536126200004 -96.079954878999956 30.09522809200007 -
96.079943789999959</gml:posList>
      </gml:LineString>
    </ location1D>
  </LinePipe>
</ PipelinMLDataset >
```

Finally, we have example 3 that lists 3 components along with point and linestring coordinate location information. These are a few simple examples of what a PipelineML file might look like. Once the PipelineML standard has been finalized, approved and published, a much more extensive set of examples will be provided. We will also be publishing a sample pipeline file populated with weld-level accurate data that will be used throughout all documents and sample files. We also anticipate developing and publishing a set of open API calls that will be helpful in exporting and importing PipelineML data from databases.

# *Glossary of Terms*

**3DIM DWG –** The 3D Information Management (3DIM) Domain Working Group is facilitating the definition and development of interface and encoding standards that enable software to develop solutions that allow infrastructure owners, builders, emergency responders, community planners, and the traveling public to better manage and navigate complex built environments. Effective integration of these software data and services has eluded the geospatial and CAD industry for decades. Today, through the cooperation of diverse stakeholders, integrated infrastructure information systems will be achieved. OGC members and partners will work in an iterative development process to achieve incremental demonstrations of real solutions. See here.

**American National Standards Institute (ANSI)** – As the voice of the U.S. standards and conformity assessment system, the American National Standards Institute (ANSI) empowers its members and constituents to strengthen the U.S. marketplace position in the global economy while helping to assure the safety and health of consumers and the protection of the environment. See here.

**American Petroleum Institute (API)** – The American Petroleum Institute (API) is the only national trade association that represents all aspects of America's oil and natural gas industry. Our more than 625 corporate members, from the largest major oil company to the smallest of independents, come from all segments of the industry. They are producers, refiners, suppliers, marketers, pipeline operators and marine transporters, as well as service and supply companies that support all segments of the industry. See here.

**American Society of Civil Engineers (ASCE)** – The American Society of Civil Engineers represents more than 150,000 members of the civil engineering profession in 177 countries. Founded in 1852, ASCE is the nation's oldest engineering society. ASCE stands at the forefront of a profession that plans, designs, constructs, and operates society's economic and social engine – the built environment – while protecting and restoring the natural environment. See here.

**American Society of Mechanical Engineers (ASME)** – ASME is a not-for-profit membership organization that enables collaboration, knowledge sharing, career enrichment, and skills development across all engineering disciplines, toward a goal of helping the global engineering community develop solutions to benefit lives and livelihoods. Founded in 1880 by a small group of leading industrialists, ASME has grown through the decades to include more than 130,000 members in 151 countries. Thirty-two thousand of these members are students. See here.

**ANSI** – *See American National Standards Institute (ANSI)*

**API** – *See American Petroleum Institute (API)*

**Appurtenance** – Things that belong to and go with something else, the appurtenance being less significant than what it belongs to. The word ultimately derives from Latin appertinere, "to appertain". See here.

**ASME** – *See American Society of Mechanical Engineers (ASME)*

**ASCE** – *See American Society of Civil Engineers (ASCE)*

**CAD –** *See Computer Aided Design (CAD)*

**Cathodic Protection** – A technique used to control the corrosion of a metal surface by making it the cathode of an electrochemical cell. A simple method of protection connects the metal to be protected to a more easily corroded "sacrificial metal" to act as the anode. The sacrificial metal then corrodes instead of the protected metal. For structures such as long pipelines, where passive galvanic cathodic protection is not adequate, an external DC electrical power source is used to provide sufficient current. See here.

**CityGML** – A common information model and XML-based encoding for the representation, storage, and exchange of virtual 3D city and landscape models. CityGML provides a standard model and mechanism for describing 3D objects with respect to their geometry, topology, semantics and appearance, and defines five different levels of detail. Included are also generalization hierarchies between thematic classes, aggregations, relations between objects, and spatial properties. CityGML is highly scalable and datasets can include different urban entities supporting the general trend toward modeling not only individual buildings but also whole sites, districts, cities, regions, and countries. See here and here.

**Comma-Separated Values (CSV)** – In computing, a comma-separated values (CSV) file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format. The CSV file format is not standardized. The basic idea of separating fields with a comma is clear, but that idea gets complicated when the field data may also contain commas or even embedded line-breaks. CSV implementations may not handle such field data, or they may use quotation marks to surround the field. Quotation does not solve everything: some fields may

need embedded quotation marks, so a CSV implementation may include escape characters or escape sequences. See here.

**Computer Aided Design (CAD)** – The use of computer systems to aid in the creation, modification, analysis, or optimization of a design. CAD software is used to increase the productivity of the designer, improve the quality of design, improve communications through documentation, and to create a database for manufacturing. CAD output is often in the form of electronic files for print, machining, or other manufacturing operations. The term CADD (for Computer Aided Design and Drafting) is also used. See here.

**Controlled Vocabulary** – A way to organize knowledge for subsequent retrieval. They are used in subject indexing schemes, subject headings, thesauri, taxonomies and other forms of knowledge organization systems. Controlled vocabulary schemes mandate the use of predefined, authorised terms that have been preselected by the designers of the schemes, in contrast to natural language vocabularies, which have no such restriction. See here.

**Coordinate Reference System (CRS)** – a coordinate-based local, regional or global system used to locate geographical entities. A spatial reference system defines a specific map projection, as well as transformations between different spatial reference systems. Spatial reference systems are defined by the OGC's Simple feature access using well-known text, and support has been implemented by several standards-based geographic information systems. Spatial reference systems can be referred to using a SRID integer, including EPSG codes defined by the International Association of Oil and Gas Producers. It is specified in ISO 19111:2007 Geographic information—Spatial referencing by coordinates, also published as OGC Abstract Specification, Topic 2: Spatial referencing by coordinate. See here.

**Coterminous** – Having the same or coincident boundaries. In this context, a series of connected components that collectively comprise a pressured pipeline system. See here.

**CRS** - *See Coordinate Reference System (CRS)*

**CSV** – *See Comma-Separated Values (CSV)*

**Domain Working Groups (DWG)** – Domain Working Groups (DWG or WG) provide a forum for discussion of key interoperability requirements and issues, discussion and review of implementation specifications, and presentations on key technology areas relevant to solving geospatial interoperability issues. See here.

**DWG** – *See Domain Working Groups (DWG)*

**Element Tag** – An XML element is everything from (including) the element's start tag to (including) the element's end tag (i.e. <price>29.99</price>). See here.

**Energy and Utilities Domain Working Group (DWG)** – The Energy & Utilities Domain Working Group will focus on the global energy and utilities community, which is defined as individuals and organizations engaged in the geospatial aspects of the planning, delivery, operations, reliability and ongoing management of electric, gas, oil and water services throughout the world. See here.

**Geography Markup Language (GML)** – The Geography Markup Language (GML) is a way to exchange geographical information between computer systems which use different software from different software providers. More specifically, GML is an XML encoding for the transport and storage of geographic information, including both the geometry and the properties of geographic features, between distributed systems. The OpenGIS® Geography Markup Language Encoding Standard (GML) The Geography Markup Language (GML) is an XML grammar for expressing geographical features. GML serves as a modeling language for geographic systems as well as an open interchange format for geographic transactions on the Internet. As with most XML based grammars, there are two parts to the grammar – the schema that describes the document and the instance document that contains the actual data. A GML document is described using a GML Schema. This allows users and developers to describe generic geographic data sets that contain points, lines and polygons. However, the developers of GML envision communities working to define community-specific application schemas [en.wikipedia.org/wiki/GML_Application_Schemas] that are specialized extensions of GML. Using application schemas, users can refer to roads, highways, and bridges instead of points, lines and polygons. If everyone in a community agrees to use the same schemas they can exchange data easily and be sure that a road is still a road when they view it. See here and here.

**Geography Markup Language Simple Features Profile (GML-SF)** – In both GML Simple Features and previous Simple Features (SF) specifications for OGC, such as Simple Features for SQL, "features are considered to be objects which can have geometry and other properties. The SF specifications are more restrictive than GML, however, in that geometry is limited to points, lines, and polygons (and collections of these), with linear interpolation between vertices of lines, and planar (flat) surfaces within polygons. The new GML Simple Features (GML-SF) profile has a similar understanding of the structure and geometry of features as SF-SQL. However, GML-SF goes beyond SF-SQL in some important ways: GML-SF supports three-dimensional coordinates (location and elevation) on feature geometry, where SF-SQL just supports two dimensions (location). GML-SF

also goes beyond SF-SQL by supporting metadata, a means of referencing local or remote resources which could be used for primary/foreign key references, and dynamic codelists. See here.

**Globally Unique Identifier (GUID)** – A globally unique identifier is a unique reference number used as an identifier in computer software. The term "GUID" typically refers to various implementations of the universally unique identifier (UUID) standard. GUIDs are usually stored as 128-bit values, and are commonly displayed as 32 hexadecimal digits with groups separated by hyphens, such as: 21EC2020-3AEA-4069-A2DD-08002B30309D. See here.

**Global Positioning System (GPS)** – The Global Positioning System (GPS), also known as Navstar GPS or simply Navstar, is a global navigation satellite system (GNSS) that provides geolocation and time information to a GPS receiver in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites.[5] The GPS system operates independently of any telephonic or internet reception, though these technologies can enhance the usefulness of the GPS positioning information. The GPS system provides critical positioning capabilities to military, civil, and commercial users around the world. See here.

**GML** – See *Geography Markup Language (GML)*

**GML-SF** – *See Geography Markup Language Simple Features Profile (GML-SF)*

**GPS** – *See Global Positioning System (GPS)*

**GUID** – *See Globally Unique Identifier (GUID)*

**IMU** – *See Inertial Measurement Unit (IMU)*

**Inertial Measurement Unit (IMU)** – An inertial measurement unit (IMU) is an electronic device that measures and reports a body's specific force, angular rate, and sometimes the magnetic field surrounding the body, using a combination of accelerometers and gyroscopes, sometimes also magnetometers. IMUs are typically used to maneuver aircraft, including unmanned aerial vehicles (UAVs), among many others, and spacecraft, including satellites and landers. Recent developments allow for the production of IMU-enabled GPS devices. An IMU allows a GPS receiver to work when GPS-signals are unavailable, such as in tunnels, inside buildings, or when electronic interference is present. See here.

**INSPIRE** – The INSPIRE Directive aims to create a European Union spatial data infrastructure for the purposes of EU environmental policies and policies or activities which may have an impact on the

environment. This European Spatial Data Infrastructure will enable the sharing of environmental spatial information among public sector organisations, facilitate public access to spatial information across Europe and assist in policy-making across boundaries. See here.

**International Organization for Standardization (ISO)** – ISO is an independent, non-governmental international organization with a membership of 163 national standards bodies. Through its members, it brings together experts to share knowledge and develop voluntary, consensus-based, market relevant International Standards that support innovation and provide solutions to global challenges. See here.

**ISO** – *See International Organization for Standardization (ISO)*

**Land and Infrastructure DWG** – The mission of the Land Development DWG is to determine the best approach for the OGC to become the standards organization responsible for the LandXML data schema. Our role is to explore the potential options and determine the most appropriate method for supporting LandXML data types within the OGC data standards framework, as well as other land information that is currently not integrated with this framework. See here.

**Land and Infrastructure SWG** – The Standards Working Group focused on defining the technical aspects of the LandInfra set of models and encoding standards. See here.

**Linear Reference System (LRS)** – A method of spatial referencing, in which the locations of features are described in terms of measurements along a linear element, from a defined starting point, for example a milestone along a road. Each feature is located by either a point (e.g. a signpost) or a line (e.g. a no-passing zone). The system is designed so that if a segment of a route is changed, only those milepoints on the changed segment need to be updated. Linear referencing is suitable for management of data related to linear features like roads, railways, oil and gas transmission pipelines, power and data transmission lines, and rivers. See here.

**LRS** – *See Linear Reference System (LRS)*

**Mill Test Report (MTR)** – A Mill Test Report (MTR) and often also called a Certified Mill Test Report, Certified Material Test Report, Mill Test Certificate (MTC), Inspection Certificate, Certificate of Test and a host of other names is a quality assurance document used in the metals industry that certifies a material's chemical and physical properties and states a product made of metal (steel, aluminum, brass or other alloys) compliance with an international standards organization (such as ANSI, ASME, etc.) specific standards. See here.

**MTR** – *See Mill Test Report (MTR)*

**PHMSA –** *See Pipeline and Hazardous Materials Safety Administration (PHMSA)*

**Pipeline and Hazardous Materials Safety Administration (PHMSA)** – The Pipeline and Hazardous Materials Safety Administration (PHMSA) operates in a dynamic and challenging environment. The scope and complexity of our safety mission will continue to grow, requiring that we fundamentally rethink how we will use data, information, and technology to achieve our safety goals. See here.

**PipelineML** – An initiative to develop an open extensible vendor-neutral international standard to enable the interoperable interchange of pipeline data between parties, disparate systems and software applications without loss of accuracy, density or data resolution; without ambiguity; and without need for conversion between intermediate or proprietary formats while maintaining contextual integrity. This standard is expected to leverage existing OGC standards such as GML, GML-SF, WFS, and WMS. The ML in the name inherits from GML (a convention commonly followed in OGC GML standards). The acronym ML is loosely defined. It can be defined as Modeling Language in terms of the conceptual model. It can also be defined as Markup Language in terms of the encoding standard. Both are applicable and hence the lack of delineation. See here.

**PipelineML SWG** – The OGC Standards Working Group responsible for developing the technical standards specification for PipelineML. See here.

**Pipeline Open Data Standard (PODS)** – The PODS Association is a recognized leader in data standards for pipeline data management, organized in 1998. It is a non-profit vendor-neutral pipeline-industry organization, member driven and volunteer-run. This dynamic and collaborative Association consists of over 170 member companies of Pipeline Operators and Service Providers. It is both created for and by the operators and providers who utilize the PODS open data structure for storing critical pipeline data and enabling analysis and reporting. The PODS Association is committed to developing, maintaining and advancing the proven PODS Pipeline Data Model, pipeline data storage and interchange standards for the pipeline industry worldwide, meeting the needs of pipeline companies today and into the future. See here.

**PODS** – *See Pipeline Open Data Standard (PODS)*

**POSC Caesar** – POSC Caesar Association (PCA) is an international, open, not-for-profit, member organization that promotes the development of open specifications to be used as standards for enabling the interoperability of data, software and related matters. PCA is the initiator of ISO 15926

"Integration of life-cycle data for process plants including oil and gas production facilities" and is committed to its maintenance and enhancement. See here and here.

**Shapefile** – The shapefile format is a popular geospatial vector data format for geographic information system (GIS) software. It is developed and regulated by Esri as a (mostly) open specification for data interoperability among Esri and other GIS software products. The shapefile format can spatially describe vector features: points, lines, and polygons, representing, for example, water wells, rivers, and lakes. Each item usually has attributes that describe it, such as name or temperature. The shapefile format is a digital vector storage format for storing geometric location and associated attribute information. This format lacks the capacity to store topological information. The shapefile format was introduced with ArcView GIS version 2 in the early 1990s. It is now possible to read and write geographical datasets using the shapefile format with a wide variety of software. The shapefile format is simple because it can store the primitive geometric data types of points, lines, and polygons. Shapes (points/lines/polygons) together with data attributes can create infinitely many representations about geographic data. Representation provides the ability for powerful and accurate computations. See here.

**SQL** – *See Structure Query Language (SQL)*

**Standards Working Group (SWG)** – Standards Working Groups (SWG) have specific charter of working on a candidate standard prior to approval as an OGC standard or on making revisions to an existing OGC standard. See here.

**Structure Query Language (SQL)** – A special-purpose domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). See here.

**SWG** – *See Standards Working Group (SWG)*

**Traceable, Verifiable and Complete (TVC)** – This terminology was utilized in the PHMSA document published on May 7, 2012 pertaining to how pipeline operators should preserve and verify records. That document states, "The advisory informs operators that records supporting MAOP and MOP should be traceable, verifiable, complete and clearly linked to original information about a pipeline segment or facility. These records must also be verified by complimentary, but separate, documentation. The advisory informs operators that to be complete, pipeline records must display a signature, date, or other appropriate marking to show the operator considers it to be a final document." See here.

**TVC** – *See Traceable, Verifiable and Complete (TVC)*

**Unified Modeling Language (UML)** – The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of software engineering, that is intended to provide a standard way to visualize the design of a system. See here.

**UML** – *See Unified Modeling Language (UML)*

**Uniform Resource Identifier (URI)** – A string of characters used to identify a resource. Such identification enables interaction with representations of the resource over a network, typically the World Wide Web, using specific protocols. Schemes specifying a concrete syntax and associated protocols define each URI. The most common form of URI is the Uniform Resource Locator (URL), frequently referred to informally as a web address. More rarely seen in usage is the Uniform Resource Name (URN), which was designed to complement URLs by providing a mechanism for the identification of resources in particular namespaces. See here.

**URI** – *See Uniform Resource Identifier (URI)*

**Web Feature Service (WFS)** – In computing, the Open Geospatial Consortium Web Feature Service Interface Standard (WFS) provides an interface allowing requests for geographical features across the web using platform-independent calls. One can think of geographical features as the "source code" behind a map, whereas the WMS interface or online tiled mapping portals like Google Maps return only an image, which end-users cannot edit or spatially analyze. The XML-based GML furnishes the default payload-encoding for transporting geographic features, but other formats like shapefiles can also serve for transport. In early 2006 the OGC members approved the OpenGIS GML Simple Features Profile. This profile is designed both to increase interoperability between WFS servers and to improve the ease of implementation of the WFS standard. See here and here.

**Web Map Service (WMS)** – A standard protocol for serving (over the Internet) georeferenced map images which a map server generates using data from a GIS database. The Open Geospatial Consortium developed the specification and first published it in 1999. The OpenGIS® Web Map Service Interface Standard (WMS) provides a simple HTTP interface for requesting geo-registered map images from one or more distributed geospatial databases. A WMS request defines the geographic layer(s) and area of interest to be processed. The response to the request is one or more geo-registered map images (returned as JPEG, PNG, etc.) that can be displayed in a browser application. The interface also supports the ability to specify whether the returned images should be transparent so that layers from multiple servers can be combined or not. See here and here.

**WFS** – *See Web Feature Service (WFS)*

**WMS** – *See Web Map Service (WMS)*

**XML Schema (XSD)** – A recommendation of the World Wide Web Consortium (W3C), specifies how to formally describe the elements in an Extensible Markup Language (XML) document. It can be used by programmers to verify each piece of item content in a document. They can check if it adheres to the description of the element it is placed in. Like all XML schema languages, XSD can be used to express a set of rules to which an XML document must conform in order to be considered "valid" according to that schema. However, unlike most other schema languages, XSD was also designed with the intent that determination of a document's validity would produce a collection of information adhering to specific data types. Such a post-validation infoset can be useful in the development of XML document processing software. See here.

**XSD** – *See XML Schema (XSD)*

## *PipelineML Contributors*

The following is a list of individuals who have contributed to the development of PipelineML as defined in this document (listed alphabetically by last name):

- Roger Abel, Shell
- Steve Adam, Adam Labs
- Thomas Adolphi, Technische Universität Berlin (Group Observer)
- Jeff Allen, Esri
- David Arctur, University of Texas at Austin (Group Observer)
- Rob Atkinson, Metalinkage, Australia
- Matt Beare, Matthew Beare (Group Observer)
- Joachim Benner, Karlsruher Institut fur Technologie
- Susan Bobbitt, Enterprise Products (GIS Consulting)
- Ron Brush, New Century Software
- Greg Buehler, Open Geospatial Consortium (OGC)
- Kurt Buehler, Image Matters LLC (Group Observer)
- Rod Burden, Enghouse, Canada (Group Member)
- Grace Cardenas, Enterprise Products (Interoperability Experiment Coordinator)
- Daniel Colby, ProStar Geocorp. Inc. (Voting Member)
- Tom Coolidge, Esri (Charter Member)
- David Danko, Esri (Group Observer)
- Lorne Dmitruk, Enbridge
- Dion Duran, Enterprise Products (Charter Member, GIS Consulting)
- Johannes Echterhoff, interactive instruments GmbH (Group Observer)
- Karim Elhanafi, Ong IT
- Vincent Fan, Sask Energy
- Peter Forster, ProStar Geocorp. Inc. (Group Observer)
- Jeanne Foust, Esri (Group Observer)
- Leif Granholm, Trimble Navigation Ltd. (Group Observer)
- Aruna Gubba, Enterprise Products
- Craig Hawkins, Crestwood Midstream Partners LP
- Karl-Heinz Haefele, Karlsruhe Institute of Technology (KIT) (Group Member)
- Rich Henry, TRC (Group Observer)
- John Herring, Oracle USA (Group Observer)
- Vincent Heurteaux, GEOMATYS (Group Observer)

- Gary Hoover, Enterprise Products (Principle Business Driver)
- Rick Hugie, Vintritech
- Terry Idol, Open Geospatial Consortium (OGC)
- David Irick, Shell Exploration & Production Company
- Johnny Jensen, Trimble Navigation Ltd. (Group Observer)
- Young-Jin Jung, Korea Land & Geospatial InformatiX Corporation (Group Observer)
- Nichole Killingsworth, BSD Consulting, Inc.
- Kyoung-Sook Kim, National Institute of Advanced Industrial Science & Technology (AIST) (Group Observer)
- Eric Klein, Global Information Systems
- Frank W. Klucznik, Georgia Technical Research Institute
- Thomas H. Kolbe, Technische Universität München - Runder Tisch GIS e.V. (Group Observer)
- Tatjana Kutzner, Technische Universität München - Runder Tisch GIS e.V. (Group Observer)
- Ron Lake, Galdos Systems Inc. (Charter Member)
- Bart De Lathouwer, Open Geospatial Consortium (OGC)
- Jeongeun Lee, Korea Land & Geospatial InformatiX Corporation (Group Observer)
- Young-Ho Lee, Shingu College (Group Observer)
- Luc van Linden, HL Conslting, Belgium (Group Observer)
- Roger Lott, International Association of Oil & Gas Producers (IOGP) (Group Observer)
- Nicolas Loubier, Bentley Systems, Inc. (Group Observer)
- Raj Mahendran, Enterprise Products
- Julie Binder Maitra, Federal Geographic Data Committee, FGDC Standards Coordinator, U.S. Geological Survey, U.S. Department of the Interior
- Lance Marrou, Leidos (Group Observer)
- Kathy Mayo, PODS Association
- Aaron Musick, Williams
- Ramani Nayak, Enterprise Products
- Michael Ortiz, New Century Software
- George Percivall, Open Geospatial Consortium, Inc., The Unified Code for Units of Measure (Group Observer)
- Perry Peterson, the PYXIS innovation (Group Observer)
- Clemens Portele, interactive instruments GmbH (Group Observer)
- James Rapaport, CARIS (Group Observer)
- Carl Reed III, Carl Reed (Group Observer)
- Jared Reid, Centurion Pipeline, L.P.

- Marcel Reuvers, Geonovum (Group Observer)
- Ian Robinson, Bentley Systems, Inc. (Group Observer)
- Nils Sandsmark, POSC Caesar Association
- Paul Scarponcini, Bentley Systems, Inc. (LandInfra DWG/SWG Chair)
- Paul Sgambati, American Society of Civil Engineers (ASCE)
- Hiren Shah, Enterprise Products
- Idan Shatz, the PYXIS innovation (Group Observer)
- Scott Simmons, Open Geospatial Consortium (OGC)
- Janet Sinclair, Retired, Formerly PODS Association
- Jason Smith, Harris Corporation (Group Observer)
- Leif Stainsby, Galdos Systems Inc. (Group Observer)
- Carl Stephen Smyth, Open Site Plan (Group Observer)
- Terry Strahan, QualityLink GIS solutions
- Jan Stuckens, Merkator NV/SA, Belgium (PipelineML SWG Co-chair)
- Trevor Taylor, Open Geospatial Consortium (OGC)
- John Tisdale, Enterprise Products (PipelineML SWG Co-chair)
- Yavuz Tor, URS Corporation
- Peter Veenstra, TRC (Group Observer)
- Warren Williamson, Bureau of Safety and Environmental Enforcement (BSEE)
- Sisi Zlatanova, Delft University of Technology (Group Observer)

# *Document Metadata*

**Publication Date**: 2016.12.02

**Version**: 1.3

**Document Type:** Introductory Knowledge-share Document

**Keywords:** PipelineML, Open Geospatial Consortium, OGC, PipelineML Standards Working Group, PipelineML SWG, Introduction, Overview, Summary, Document, Documentation

**Author**: John Tisdale

**Biography**: John Tisdale has over 30 years of experience as a technology solutions architect. He has worked extensively throughout Silicon Valley helping technology firms develop more advanced technologies to solve complex business challenges.

### Change Tracking Log

Version 1.3, 12.2.2016, J Tisdale – Changed the language of OGC certification to OGC approval.

Version 1.2, 12.1.2016, J Tisdale – Corrected formatting and refined language.

Version 1.1, 11.30.2016, J Tisdale – Several changes were made to the content based on feedback from stakeholders pertaining to the support of linear referencing and coordinate referencing, changed language used to describe spatial accuracy support, added new terms to glossary.

Version 1.0, 11.29.2016, J Tisdale – This is the official release of the document.