Open Geospatial Consortium Inc.

Date: 2006-03-03

Reference number of this document: OGC® 06-010r2

Version: 1.0.0

Category: OpenGIS® Implementation Specification

Editor: Steve Havens

OpenGIS®

Transducer Markup Language

Implementation Specification

Warning

This document is not an OGC Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type:	OpenGIS® Implementation Specification
Document subtype:	Engineering Specification
Document stage:	Request for Comment
Document language:	English

Contents

Page

Figures

Page

Tables

Page

i. Document terms and definitions

This document uses the specification terms defined in Subclause 5.3 of [OGC 05-008], which is based on the ISO/IEC Directives, Part 2, "Rules for the structure and drafting of International Standards". In particular, the word "shall" (not "must") is the verb form used to indicate a requirement to be strictly followed to conform to this specification.

ii. Submitting organizations

The following organizations submitted this document to the Open Geospatial Consortium Inc.

1) 3eTI
2) Intergraph
3) Innovative Research Ideas & Services Corporation
4) National Geospatial-intelligence Agency
5) Oak Ridge National Laboratory
6) Radiance, Inc.
7) SeiCorp, Inc.
8) York University

iii. Document contributor contact points

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

CONTACT	ORGANIZATION	ADDRESS
Steve Havens (editor)	IRIS Corporation	4220 Varsity Dr. Suite E Ann Arbor, MI 48018
Arthur Na	IRIS Corporation	4220 Varsity Dr. Suite E Ann Arbor, MI 48018
Jon Schlueter	IRIS Corporation	4220 Varsity Dr. Suite E

		Ann Arbor, MI 48018
Rachel Weingrad	IRIS Corporation	4220 Varsity Dr. Suite E Ann Arbor, MI 48018
Valerie Yoder	IRIS Corporation	4220 Varsity Dr. Suite E Ann Arbor, MI 48018
Don Jenkins	IRIS Corporation	4835 University Square, Suite 15 Huntsville, AL 35816
Kurt Steele	Kurt Steele	225 E82nd, New York NY 10028
Maj. Dan Bagley	National Geospatial-intelligence Agency	Acquisition Directorate, T579, MS P-102 Airborne Projects Office 12310 Sunrise Valley Drive Reston, VA 20191-3449
Dr. Johnny Tolliver	Oak Ridge National Laboratory	P.O. Box 2008 M/S 6085 Oak Ridge, TN 37831
Kirk Deweese	Radiance Technologies	350 Wynn Drive, Huntsville AL 35805
Jim Greenwood	SeiCorp, Inc.	5870 Trinity Pky, Suite 350 Centerville, VA 20121
Bill Craig	SeiCorp Inc.	5870 Trinity Pky, Suite 350 Centerville, VA 20121
Mark Priest	3eTI	700 King Farm Blvd. Suite 600 Rockville, MD 20850
Gene Grindstaff	Intergraph, Inc.	P.O. Box 6695 Huntsville, AL 35824
Dr. Mike Botts	University of Alabama, Huntsville	ESSC / NSSTC Huntsville, AL 35899
Steve Liang	York University	4700 Keele St. Toronto, ON M3J1P3

iv. Additional Contributors

9) Rockwell-Collins

10) Air Force Research Laboratory/Sensors Directorate

11) Lockheed-Martin

12) US Special Operations Command (USSOCOM)

13) General Dynamics

14) Khoral

15) Modus Operandi

16) SSAI

17) SAIC

18) Northrop Grumman

v. Revision history

Date	Release	Author	Section modified	Description
2003-08-25	v0.095beta	swh	all	Original TML document
2005-10-01	v1.0.0	swh	all	Reformatted document into OGC Implementation Specification format
2006-02-28	v1.0.0	swh	all	All sections revised and updated

vi. Changes to the OGC Abstract Specifications

The OpenGIS® Abstract Specification does not require changes to accommodate the technical contents of this document.

vii. Future work

Areas of future work include addressing new and expanded applications above and beyond geospatial applications. TML implementations will refine and clarify existing components within TML as well as providing more examples of how transducers and transducer systems are modelled.

Foreword

This edition of the OGC TML Implementation Specification replaces all previous (beta) TML specifications. TML was developed under a government contract to enable the interoperability and fusion of dissimilar sensor data. Organizations contributing to this work are identified in section iii Submitting Organizations, section iv Document Contributors, and section v Other Contributors. Section v titled "Other Contributors", indicates organizations that contributed to the development of TML but are not OGC members.

Prior to being submitted to OGC, TML was developed and demonstrated with its roots in the sensor community. NGA requested that OGC investigate the adoption of TML for the exchange and archive of streaming sensor data. During OWS-3 it was demonstrated how TML would enable the capability to discover, describe, and exchange streaming sensor data using the OGC Sensor Observation Service and the Sensor Alert Service. TML fits very well into the SWE family of services for the exchange and archive of streaming sensor data.

TML is designed to work within transducer exchange messages from multiple application domains (defense, weather, exploration, environmental, medical, industrial, security, etc.). To provide a complete picture within these messages, the TML sensor data description will be complemented with other domain specific information. For example in the defense domain, specific information would include security, bombing encyclopedia number, sortie and mission numbers, etc. However, this information would not be applicable in the medical domain.

This document includes four annexes; Annex A is informative, Annexes B, C and D are normative.

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

For additional information on TML visit http://www.transducerml.org.

Introduction

Transducer Markup Language (TML) defines:

- a set of models describing the hardware response characteristics of a transducer.
- an efficient method for transporting sensor data and preparing it for fusion through spatial and temporal associations.

Sensor data is often an artifact of the sensor's internal processing rather than a true record of phenomena state. The effects of this processing on sensed phenomena are hardware-based and can be characterized as functions.

TML response models are formalized XML descriptions of these known hardware behaviors. The models can be used to reverse distorting effects and return artifact values to the phenomena realm. TML provides models for a transducer's latency and integration times, noise figure, spatial and temporal geometries, frequency response, steady-state response and impulse response.

Traditional XML wraps each data element in a semantically meaningful tag. The rich semantic capability of XML is in general better suited to data exchange rather than live delivery where variable bandwidth is a factor. TML addresses the live scenario by using a terse XML envelope designed for efficient transport of live sensor data in groupings known as TML clusters. It also provides a mechanism for temporal correlation to other transducer data.

OpenGIS® Transducer Markup Language Implementation Specification

1 Scope (Informative)

This document describes TML and how it captures necessary information to both understand and process transducer data. Descriptions of how TML captures the actual data and also descriptions of necessary information such as system calibration, transducer behavior, conditions of operation and data collection parameters critical to logical data structure model are all captured within TML. This document specifically describes the TML transducer behavior models and TML data models. It does not describe a service for delivering TML-structured data, but there is a brief discussion of SOA and streaming platform considerations.

2 Conformance (Informative)

Most of the schema components specified in this International Standard implement concepts defined in the ISO 19100 series of International Standards. In these cases, the conformance classes defined in this International Standard are based on the conformance classes defined in the corresponding standard.

Any TML Application Schema or software implementation claiming conformance with one of the conformance classes shall pass all test cases of the corresponding abstract test suite.

3 Normative references (Normative)

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies. In the event of a conflict between the text of this document and the references cited herein, the test of this document takes precedence. Nothing in this document, however, supersedes applicable laws and regulations unless a specific exception has been obtained.

DMA TR 8358.1 *Datums, Ellipsoids, Grids, and Grid Reference Systems*

IEEE 100, *The Authoritative Dictionary of IEEE Standards Terms*

IEEE Std 754-1985 (Reaff 1990), *IEEE Standard for Binary Floating-Point Arithmetic.*

ISO 1000 *SI Units and Recommendations for the use of Their Multiples and of Certain Other Units*, 1992

ISO 10646-1:1983 Amd2 1986 *Information Technology – Universal Multiple - Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane – Attachment 2: UCS Transformation Format 8 (UTF-8)*

ISO/IEC CD 18026 *Information technology – Computer Graphics and Image Processing – Spatial Reference Model*

ISO 19108 *Geographic Information – Temporal schema*

ISO CD 19130 *Geographic Information – Sensor and Data Model for Imagery and Gridded Data*

NIMA TR8350.2 *Department of Defense World Geodetic System 1984*, Third Edition 4 July 1997

OGC 03-040 *OpenGIS® Reference Manual*, 4 Mar 2003

OGC 04-019r2, *Sensor Model Language, (SensorML),* 1 Nov 2004

OGC 04-071 *Some Image Geometry Models*, 30 Sep 2004

OGC 05-008, *OpenGIS® Web Services Common Specification*

OGC Abstract Specification, Topic 2 - *Spatial Referencing by Coordinates. Topic 2*

Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation (6 October 2000)

*Namespaces in XM*L. W3C Recommendation (14 January 1999). Available [Online]: http://www.w3.org/TR/1999/REC-xml-names-19990114

*XML Schema Part 1: Structure*s. W3C Recommendation (2 May 2001). Available [Online]: http://www.w3.org/TR/xmlschema-1

*XML Schema Part 2: Datatype*s. W3C Recommendation (2 May 2001). Available [Online]: http://www.w3.org/TR/xmlschema-2/

In addition to this document, this specification includes several normative XML Schema Document files as specified in Annex B.

4 Terms and definitions (Normative)

This section contains key terms and definitions required to understand the TML concepts.

For the purposes of this specification, the definitions specified in Clause 4 of the OWS Common Implementation Specification [OGC 05-008] and in OpenGIS® Abstract Specification Topic 2: *Spatial referencing by coordinates* shall apply. In addition, the following terms and definitions apply.

4.1 actuator

A subdivision of transducrs. An actuator transforms input data into an in-situ action or phenomenon.

4.2 attitude invariant transducer

An attitude invariant transducer will exhibit the same response independent of the attitude or orientation it is in. In-situ transducers, which do not have a direction associated with them, and remote transducers with an omni-directional ambiguity space are attitudinally invariant.

4.3 baseband signal

Original signal to or from a transducer prior to modulation onto a carrier or sub-carrier, or after demodulation from a carrier or sub-carrier signal.

4.4 coordinate system

A system or methodology for positioning points within a spatial reference system. TML utilizes only three coordinate systems: rectangular, cylindrical, and spherical.

4.5 cylindrical coordinate system

3-dimensional coordinates with one angle coordinate (alpha) and two distance coordinates (r, z), r is the length of the line between the normal projection of the point onto the xy plane and the origin. The two dimensional version of the cylindrical coordinate will be referred to as a polar coordinate system utilizing only the α and r coordinate.

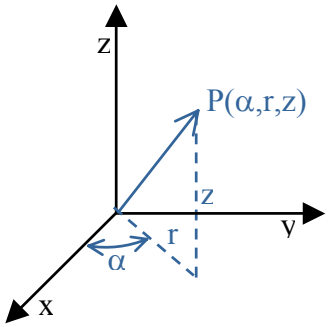


Figure 1 - Cylindrical coordinate system

4.6 data description/metadata

The data description is a self-contained description of the TML data clusters with information about the transducer, its behavior, information pertaining to the conditions of operation which might be relevant to later processing of data, and the layout of the data

within the TML data stream. This comprises all information contained in the system, transducer, relations, process and data cluster tags, so that a complete picture of the data and how to process it can be formed.

4.7 data group

A group of like data units. When many similar data units are described in TML it is efficient to factor out the common factors describing each data unit in the group. Then the description of each individual data unit only needs to describe it differences. The propose of the data group is to shorten the length of the data description. This becomes important for example when you have 10,000 spectral bands you may need to characterize.

4.8 data index

A count or count to uniquely identify each data set (e.g. pixel) location with a unique index or count with a data array structure (e.g. image frame). The data array structure may be defined with up to three indexes related to row, column, and plane, depending in the order of transducer scanning through the structure. Row increments first, column increments second, and plane increments last. If no data structure is defined, the data array will be treated as a 1D structure. Data indexing shall use the following scheme related to the dimensions in the data structure, for example using i, j, and k to represent the index of the array indexing would follow the following pattern. Index for a 1D (i) array, indexing for a 2D array (i,j), indexing for a 3D array (i,j,k). Indexing for nested data arrays has the parent index before the child index, for example for a single nested array using l, m, and n to represent the index of the parent array. 1D parent array of 1D child arrays (l,i), 1D parent array of 2D child arrays (l,i,j), 1D parent array of 3D child arrays (l,i,j,k), 2D parent array of 1D child arrays (l,m,i), 2D parent array of 2D child arrays (l,m,i,j), 2D parent array of 3D child arrays (l,m,i,j,k), 3D parent array of 1D child arrays (l,m,n,i), 3D parent array of 2D child arrays (l,m,n,i,j), 3D parent array of 3D child arrays (l,m,n,i,j,k), and so on for higher nesting levels. This represents the logical structure as the transducer data should be organized. During the serialization and buffering process, this order or sequence may be disturbed. For describing re-sequencing of data the indexing shall be flattened to a single coordinate such as:

```
data index number=0
for l=1 to num col of parent
    for m=1 to num rows of parent
        for n=1 to num planes of parent
            for i=1 to num col of child
                for j=1 to num rows child
                    for k=1 to num planes child
                    increment data index number
```

This single coordinate data index number can then be re-sequenced to represent any resorting of data required during the serial exchange of data.

4.9 data product

Data products refer to images, reports, analysis and other information produced as a result of processing raw transducer data. Data products may be in formats other than TML and the TML description can support describing other transducer formats. However, the accuracy and understandability of the data is not assured.

4.10 dimensionality

Data captured in a transducer characteristic frame is said to have n-dimensions (or dimensionality). The number of dimensions are determined by the number of spatial coordinates used to define the spatial geometry of the data. Dimensionality is of the internal geometry of the data only. Time is not considered a dimension. Although a multi-dataSet TCF may have only temporal coordinates and no spatial coordinates associated with each dataset in the TCF, it is still a non-dimensional TCF. An in-situ does not have an internal spatial coordinates, therefore it has a dimensionality of zero. For example, data that uses a table of alpha and beta coordinates will have a dimensionality of two.

4.11 earth centered earth fixed (ECEF) spatial reference system

The origin of this reference system is located at the center of mass of the earth. The right hand orthogonal axis x, y, and z are oriented with x at the prime meridian and z north. Objects in the ECEF Spatial Reference System may be positioned using either the rectangular, spherical, or the WGS-84 geodetic Coordinate System

NOTE: refer to WGS-84

4.12 ellipsoidal heightgeodetic /height /altitude/ elevation

The elevation or altitude is the distance of a point from the surface of the *reference ellipsoid* along the direction of a vector normal to the ellipsoid surface.

NOTE 1 See ellipsoidal height and gravity-related height. It should be noted that ellipsoidal height is defined w.r.t. an ellipsoidal model of the shape of the earth. Ellipsoidal height is measured from the point along the line perpendicular to the ellipsoid's surface. Also, it should be noted that gravity-related height is defined w.r.t. the geiod model of gravity. Gravity-related height is measured from the point along the line perpendicular to the geoid surface.

NOTE 2 Height or altitude of a point outside the surface is treated as positive; negative height is also named as depth. This is an exception that needs to be noted because the local reference system xyz axes are aligned with north (x), east (y), and down (z).

4.13 Euler angles

Three rotation angles (ω,φ,κ) describing how a child spatial reference system is oriented relative to a parent spatial reference system. When describing the child orientation relative to the parent the angles are derived in the following order, κ,φ,ω (i.e. heading, pitch, and roll). When describing the parent orientation relative to the child the angles are derived in the following order, ω,φ,κ (i.e. roll, pitch, and heading).

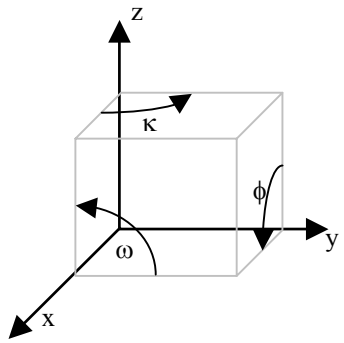


Figure 2 - Euler angles

4.14 geocentric coordinate reference system

A spherical reference system based using the ECEF spatial reference system. The coordinates used to specify a point are: geocentric latitude, longitude and radius. The radius is the distance from the origin of the ECEF reference frame to the point. Longitude is the same as the alpha angle in the spherical system, latitude is the same as the beta angle in the spherical system, and radius is the same as rho in the spherical system.

4.15 geocentric latitude

The geocentric latitude coordinate is the angle between a line defined by a point on the earth surface and the center of the ECEF reference system and the equatorial plane. The geocentric latitude is the same as the beta angle within the ECEF reference system.

4.16 geodetic coordinate reference system/ellipsoidal coordinate system

A method for describing the three dimensional position of a point within the ECEF spatial reference system. The coordinates used are geodetic latitude, longitude, and geodetic altitude. The coordinates are defined using elliptical coordinates; see the definition for geodetic latitude, longitude, and geodetic altitude).

4.17 geodetic latitude/ latitude/ ellipsoidal latitude

The geodetic latitude coordinate is the angle between the normal (line perpendicular to) to the earth reference ellipsoidal surface and the equatorial plane.

4.18 Gregorian calendar

Calendar in general use first introduced in 1582 to correct an error in the Julian calendar.

NOTE In the Gregorian calendar, common years have 365 days and leap years 366 days divided into 12 sequential months and is designed to keep synchronization with the tropical year.

4.19 image geometry model

Mathematical model that specifies the mapping (or projection) from 3D ground position coordinates to the corresponding 2D image position coordinates or visa-versa. An example image geometry model is the Replacement Sensor Model (RSM), and the Rational Polynomial Coefficient (RPC) model.

NOTE 1 The TML geometry model may not map directly to 3D ground coordinates. The TML geometry model provides a mapping from the data index to a spatial ambiguity defined relative to some spatial reference system (ECEF, local, or transducer). To determine the 3 dimensional discrete locations of data the ambiguity space of the data (e.g. IFOI) must be intersected with other ambiguity spaces (other IFOI perspective or terrain model).

NOTE 2 An image geometry model is alternately called an image sensor model, sensor model, imaging model, or image mathematical model. The term "sensor" is often used when the image is generated by a digital camera and is thus originally digital. The word "camera" is usually used when the image is recorded in analogue form, normally on film. Of course, film images can be later scanned or digitized and are then "digital".

NOTE 3 An image geometry model can also be used to determine the correct ground position for an image position, if used with additional data. When a single (or monoscopic) image is used, this additional data normally defines the shape and position of the visible ground (or object) surface. For example, this additional data is often a single elevation or is grid elevation data, sometimes called a Digital Terrain Model (DTM). Alternately, two stereoscopic images or multiple overlapping images can be used, that show the same ground point viewed from different directions. In this case, the two (or more) image geometry mathematical models can also be used, with the point coordinates in each individual image, to determine the corresponding 3D ground position.

4.20 latitude/geodetic latitude/ellipsoidal latitude

Angle from the equatorial plane to the perpendicular to the ellipsoid through a given point, with northwards being treated as positive.

4.21 linear time invariant system/ LTI system

A linear time invariant transducer or process is both time invariant and linear. A transducer or a process is said to be time invariant if a time shift in the input results in the same time shift of the output. A transducer or a process is said to be linear if it exhibits the criteria for superposition. Superposition requires that the relation between the input and the output have two properties, additivity and homogeneity. Additivity implies that if an input a results in output a' and input b results in output b', then input $a+b$ results in output $a'+b'$. Homogeneity implies that if input a results in a', then input $k(a)$ results in $k(a')$, where k is a constant.

4.22 linearity

A property of the steady state transfer function. The linearity of a transducer is described in terms of percentage of how far the response deviates from a perfect linear response over the operating range of the transducer.

4.23 live/ historical (antonym)

This term when used in association with transducer data implies that something is happening at the same time or very close to the same time as original real world phenomena or events. It has to do with the latency time between real world events and using the data. Data can be processed or viewed live or you can process or view archive data. Viewing transducer data would be considered live if the display were an accurate representation of the current events of phenomena. If the playback or viewing of transducer data represents data from a past time, then this would be historical data.

4.24 local earth spatial reference system

The origin of this reference system is located on the surface of the earth (WGS-84 ellipsoid model). If the coordinates are to be references in other local references or relative to ECEF reference, then the local earth spatial reference system must be qualified with a specified geodetic latitude and longitude coordinate specifying the origin. The right hand orthogonal coordinate axis x, y, and z are oriented in the north, east, and down direction respectively. Objects in the local earth spatial reference system can be positioned using either the rectangular or spherical coordinate system

4.25 location invariant transducer

A location invariant transducer will exhibit the same response independent of the location it is in.

EXAMPLE: Sensors which measure their attitude relative to a spatial reference system are location invariant. The location of that sensor is unimportant, only the attitudinal relationships are.

4.26 Longitude/geodetic longitude/geocentric longitude/ellipsoidal longitude

Angle from the prime meridian plane to the meridian plane of the given point, with eastward treated as positive.

NOTE: The geodetic longitude is the same as the geocentric longitude which is the same as the α coordinate in the ECEF Reference System using the spherical coordinate system.

4.27 modulation

The process of putting a lower frequency in a higher frequency signal. There are several processes for modulation such as amplitude, frequency, phase, and variants thereof.

4.28 Phenomenon/physical phenomenon

A phenomenon is a physical property that can be observed and explained by physics. Phenomenon relate to properties of matter, energy, and space-time. A phenomenon can be observed by a sensor or changed with an actuator. Table 1 - Phenomenon and phenomenon properties gives examples of various phenomenon and their properties. A phenomenon may be spatially and/or temporally modulated. In other words, its value

may change as a function of location and time [f(x,y,x,t)]. Picture images are created by representing the spatial modulation. Phenomenon can be artificially created. The phenomenon generated [f(g(x,y,z,t))].by artificial means may be a function of the input data to a transmitter or actuator [g(x,y,z,t)].

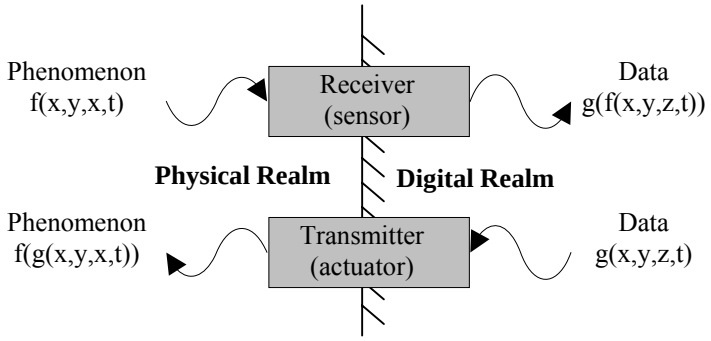


Figure 3 - Phenomenon relating to data

As part of understanding what transducer data is, the proper understanding of the phenomena and phenomena properties is critical. The following list is only a small sample of the phenomena and properties commonly used. OGC shall maintain a library of phenomena and phenomena properties.

Table 1 - Phenomenon and phenomenon properties

Phenomena	Phenomena Properties
Acoustic	Wave Amplitude, phase, polarization, Spectrum, Wave velocity
Electric	Charge, Current, Potential, Resistance, Conductivity, Capacitance, Inductance, Permittivity, Electrical Field Amplitude, Phase, Polarization, Spectrum
Magnetic	Magnetic Field, Amplitude, Phase, Polarization, Spectrum, Magnetic Flux, Permeability
Optical	Wave amplitude, phase, polarization, spectrum, Wave velocity, Refractive Index, Emissivity, Reflectivity, absorption
Mechanical	Position (linear, angular), Velocity, Acceleration, Force, Work, Power, Stress, Pressure, Strain, Mass, Density, Moment, Torque, Rate of mass transport (flow rate), Shape, Roughness, stiffness, Hardness
Thermal	Temperature, Heat, Flux, Specific Heat, Thermal conductivity, Enthalpy
Radiation	Type, Energy, Intensity
Biological	Biomass type, concentration, state
Chemical	Component identity, p-h, chemical potential, ionization rate, rate of reaction, molecular weight, concentration, state

4.29 Positional invariant transducers

A positional invariant transducer may be either locationally invariant or attitudinally invariant. A non-positional invariant transducer is neither locationally or attitudinally invariant. If a transducer is locationally invariant then the location of the transducer is unimportant. If a transducer is attitudinally invariant then the attitude of the transducer is unimportant.

EXAMPLE: If an in situ temperature sensor measures the same temperature independent of what orientation it is in then this transducer is attitudinally invariant. This would imply that there is no shaped field (i.e. ambiguity space) around the transducer. It must be either omni-directional or truly a point (in situ) detector.

EXAMPLE: If we know transducer 1 is oriented in a particular orientation relative to transducer 2, and positioned relative to transducer 3, and transducer 2 measures it's own orientation relative to an absolute datum, then to determine the attitude of transducer 1 the position of transducer 2 is unimportant. The fact that the orientation of transducer 1 is fixed relative to transducer 2 implies a ridged body and every point on that rigid body experiences the same angular rotations, therefore the position of the attitudinal measurement form transducer 2 is unimportant.

4.30 real-time
less than real-time
greater than real-time

This term refers to the playback or viewing rate of transducer data. For example if data took one minute to collect and the viewing or playback time took one minute, then this would be considered real-time. It is independent of *live* or *archive* data. Obviously viewing live data is typically real-time unless the viewing is slowed down, then it becomes archive data after a short time. If data viewing is slowed then it becomes less than real-time. If data viewing is sped up then it becomes greater than real-time. A distinction should be drawn between real-time and live; real-time refers to the rate of data playback while live refers to data playback at the same time that it is captured.

4.31 rectangular coordinate system
cartesian coordinate system

A set of orthogonal coordinates specifying the location of a point relative to a spatial reference point (origin). The coordinates are specified by normal projections of the point onto the three cardinal axes (x, y, & z). The displacements of these projections represent the three rectangular coordinates for specifying the three dimensional location of the point. The rectangular coordinates are represented by specifying the linear displacement of the intersections of the normal projections from the point and the three cardinal axes.

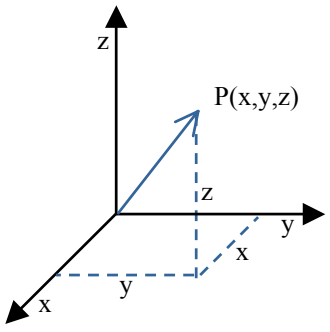


Figure 4 - Rectangular coordinate system

4.32 reference ellipsoid

An ellipsoid used as the best local or global approximation of the surface of the geoid. The reference ellipsoid is the reference surface for calculating the geodetic latitude and geodetic height.

4.33 saturation

The exact point of the response model of an LTI system where increasing the phenomenon no longer corresponds to an increase in the data value

4.34 sensitivity

The minimum level of a phenomenon's property to which data can correspond.

4.35 sensor receiver

a specific type of transducer for detecting and measuring a phenomenon. A transducer which converts a physical phenomenon into a digital data representation.

4.36 SensorML

See OGC document OGC 04-019r2, *Sensor Model Language (SensorML),* 1 Nov 2004

4.37 spatial coordinate reference system

A spatial system containing both a spatial reference system and a coordinate reference (datum) system

4.38 spatial coordinate system
coordinate system
The methods by which a three dimensional point is can be specified within a spatial reference system. The various coordinate systems are: rectangular coordinate system (see definition), spherical coordinate system (see definition), geodetic coordinate system (see definition), and the cylindrical coordinate system (see definition).

4.39 spatial invariance

Spatial invariance relates to internal geometry models. An internal geometry model must be spatially invariant, meaning the spatial characteristics of the model are independent of the location or attitude of the transducer anywhere in the world.

4.40 spatial reference system
spatial datum

This specifies the origin and attitude of the right hand orthogonal spatial coordinate axis frame. Different spatial reference (datum) systems are:

- 1) ECEF spatial reference system (see definition),
- 2) earth local spatial reference system (see definition),
- 3) transducer spatial reference system (see definition)

4.41 spherical coordinate system

1) the first angular coordinate alpha (α) is derived by the angle of rotation of the xz plane about the z axis until the plane contains the point. The second angle beta (β) is derived by the angle of rotation of the rotated xy plane about the y axis until the plane contains the point. The third length coordinate rho (ρ) is the distance from the point to the origin along a line containing both the point and the origin. 2) a spatial coordinate specified by two (2) angle coordinates and one (1) length coordinate relative to a spatial reference system.

NOTE Not to be confused with an ellipsoidal coordinate system based on an ellipsoid 'degenerated' into a sphere.

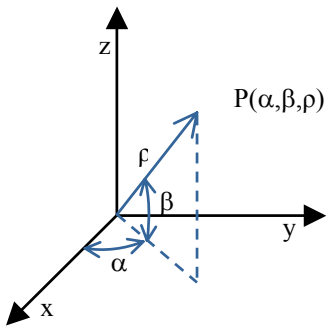


Figure 5 - Spherical Coordinate System

4.42 standard units of measure
units of measure
UOM

The defined quantity in which dimensioned parameters are expressed.

Table 2 - Standard Units of Measure

Dimension	Unit	Symbol
Length	meter	m
Mass	Kilogram	Kg
Time	second	s
Electric current	ampere	A
Temperature, thermodynamic	kelvin	K
Amount of matter	mole	mol
Luminous intensity	candela	cd
Plane angle	radian	rad
Solid angle	steradian	sr
Year of Date	year (Gregorian)	YYYY
Month of Year of Date	month(Gregorian)	MM (1-12)
Day of Month of Year of Date	day(Gregorian)	DD (1-31)
Hour of Day of Date	hour (UTC)	HH (0-23)
Minutes of Hour of Day of Date	minutes (UTC)	MM (0-59)
Seconds of Minute of Hour of Day of Date	seconds (UTC)	SS

4.43 streaming data

Data which has a time component to it. Data within the stream may occur at different instances in time. The TML data stream time stamps each data cluster so that the data can be utilized in the proper time sequence. The data may be live or archive. If the data is live, the end of the file is not yet known. Applications should not wait until the end of file to utilize the live data. If the data is from an archive then obviously the file length is known.

4.44 temporal coordinate reference system

A temporal measurement system which contains references to both a temporal reference (datum) and a temporal coordinate systems (coordinate units).

4.45 temporal coordinate system

A set of one or more coordinates specifying the time of an event relative to a temporal reference (datum) system. Common temporal coordinate systems include the Gregorian calendar, with: years, months, days, hours, minutes, and seconds.

4.46 temporal invariance

Relating to internal geometry models. An internal geometry model must be temporally invariant. This means the internal temporal characteristics of the model are independent of the time and date. The temporal characteristics are defined relative to an internal temporal reference system such that the temporal characteristics are temporally invariant to external temporal reference systems. The temporal characteristics look the same on Monday as they do on Tuesday. They appear identical regardless of time.

4.47 temporal reference system
temporal datum

The reference event to which an event is related to in time. Either the reference can be an enterprise *absolute* temporal reference, such as the 1970 Epoch, or it may be a *relative* reference such as the time of a transmitter trigger pulse, to which received data is time referenced. Common reference systems are the Gregorian calendar reference in which we are in the year 2005, UTC time reference in Greenwich UK, and the Epoch on 1 Jan 1970.

4.48 TML
Transducer Mark-up Language

A transducer exchange language described in this specification

4.49 TML application (processor/controller)

A TML processor is a computing machine to understand and process the TML message, a TML controller is a computing machine to generate TML messages. A single computing device may contain one or both functions. (See **Figure 7** - Generic TML implementation) A Processor/Controller can also be a process node on a network - providing services to clients on the network.

4.50 TML application node

A node presents TML data on a network for TML applications. It may use a set of common software libraries that enable a user application to ingest, process, display, and create (control) TML data. (See **Figure 7** - Generic TML implementation)

4.51 TML network node

A network node for the distribution of live and/or historical TML data. The network node may federate multiple network, transducer, and process nodes. (See **Figure 7** - Generic TML implementation)

4.52 TML process node

A process node presents TML processes on a network for process systems. This node may contain adapters for transducers, processes, system clock, network, archive, and registry. The transducer node will manage and maintain data description for the data it is handling, and system clock and timing. (See **Figure 7** - Generic TML implementation) The process node will input TML data and output processed TML data.

4.53 TML transducer node

A node which presents transducer information on a network for transducer systems. This node may contain adapters for transducers, processes, system clock, network, archive, and registry. The transducer node will manages and maintain data description for the data it is handling, and system clock and timing. (See **Figure 7** - Generic TML implementation)

4.54 transducer adapter

An adapter that provides the interfaces between a transducer and the TML transducer node. A unique adapter will need to be built for each unique transducer interface. This puts any proprietary or unique interface behind the TML transducer node such that the proprietary or uniqueness of the interface is transparent to the rest of the network. The TML adapter will contain the TML data description if it does not come from the transducer. The adapter will contain the timing and synchronization processing to ensure the proper time relations exist between the data and the phenomenon. For example, the time stamped on sensor data needs to represent the time of the phenomenon instance and not the time the data is outputted. For transmitters and actuators the application of the phenomenon instance needs to occur at the precise time of the indicated time tag or as soon as possible in the absence of a time tag. The adapter will group transducer data into data clusters for receivers or sensors. IEEE P1451 Smart transducer interface will greatly simplify this task and make the number of adapters required to be developed much less. Only one adapter will need to be built for all IEEE 1451 transducers. (See **Figure 7** - Generic TML implementation)

4.55 transducer data event
transducer event

An event (usually the result of an analog to digital conversion trigger) which samples the state of a sensing transducer or an event that commands an actuator. A trigger source can provide periodic triggers or it can provide a trigger only when certain criteria are met such as a threshold condition. An event will have a location and a time associated with it.

4.56 transmitter

A subdivision of transducers. A transmitter transforms input data into a remote action or phenomenon. Normally thought of in conjunction with radio electro-magnetic energy, but can apply to other propagating energy forms (e.g. acoustic).

4.57 UTC - Coordinated Universal Time

Time scale maintained by the Bureau International des Poids et Mesures (International Bureau of Weights and Measures) and the International Earth Rotation Service (IERS) that forms the basis of a coordinated dissemination of standard frequencies and time signals

4.58 WGS-84 geodetic coordinate system

Coordinate system in which position is specified by geodetic latitude, geodetic longitude and (in the three-dimensional case) ellipsoidal height, associated with the ECEF spatial reference system. The geodetic latitude is the angle between the equatorial plane and a line perpendicular to the ellipsoid surface in the plane of a constant meridian longitude. The origin and the ellipsoidal parameters (semi-major and semi-minor axis) are specified in the WGS-84 documentation.

NOTE 1 see ellipsoidal or geodetic coordinate system in OGC Abstract specification Topic 2.

NOTE 2 see geodetic latitude. geodetic latitude ≠ geocentric latitude (spherical β angle)

5 Conventions (Informative)

5.1 Normative Verbs

In the sections labeled as normative within this document, the key words "must", "must not", "required", "shall", "shall not", "should", "should not", "recommended", "may", and "optional" are to be interpreted as described in Internet RFC 2119 [1].

5.2 Abbreviated terms

Most of the abbreviated terms listed in Subclause 5.1 of the OWS Common Implementation Specification [OGC 05-008] apply to this document, plus the following abbreviated terms.

Table 3 - Abbreviated Terms

Abbreviation	Term
API	Application Programming Interface
COTS	Commercial Off The Shelf
CGI	Common Gateway Interface
CRS	Coordinate Reference System
CS	Coordinate System
DCP	Distributed Computing Platform
DTD	Document Type Definition
EPSG	European Petroleum Survey Group
GIF	Graphics Interchange Format
GIS	Geographic Information System
GML	Geography Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
ISO	International Organization for Standardization
IETF	Internet Engineering Task Force
MIME	Multipurpose Internet Mail Extensions
OGC	Open GIS Consortium
OWS	OGC Web Service
SPS	Sensor Planning Service
SSL	Secure Socket Layer
SWE	Sensor Web Enablement
URL	Uniform Resource Locator
WGS	World Geodetic System 84
XML	Extensible Markup Language

5.3 UML notation

Most diagrams that appear in this specification are presented using the Unified Modeling Language (UML) static structure diagram, as described in Subclause 5.2 of [OGC 05-008].

The diagrams that appear in this document are presented using the Unified Modeling Language (UML) static structure diagram. The UML notations used in this document are described in the diagram below.

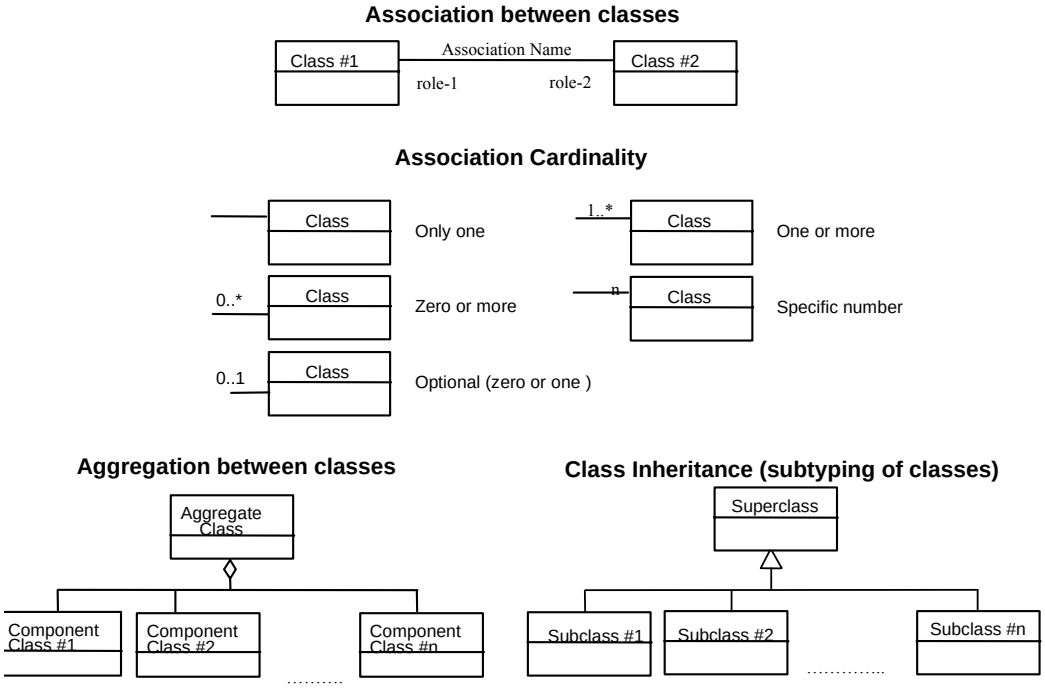


Figure 6 - UML Notation

In this diagram, the following three stereotypes of UML classes are used:

a) <<Interface>> A definition of a set of operations that is supported by objects having this interface. An Interface class cannot contain any attributes.

b) <<DataType>> A descriptor of a set of values that lack identity (independent existence and the possibility of side effects). A DataType is a class with no operations whose primary purpose is to hold the information.

c) <<CodeList>> is a flexible enumeration that uses string values for expressing a list of potential values.

In this document, the following standard data types are used:

a) CharacterString – A sequence of characters

b) Integer – An integer number

c) Real A real number

d) Boolean – A value specifying TRUE or FALSE

e) DateTime – A character string as specified by ISO 19108, which comprises year, month, day and time of the day to the appropriate level of precision. In addition, a Sequence type of collection is used, which contains an ordered list of values with the specified data type. The format used is "Sequence<DataType>".

5.4 Platform-neutral and platform-specific specifications

The TML specification is platform neutral. There are no platform specific requirements.

5.5 Italic terms

Italic terms in this document identify that the term is defined in section 4. Terms and definitions

5.6 XML schema notation

Most diagrams that appear in this specification are presented using an XML schema notation defined by the XMLSpy[1] product and described in this subclause. XML schema diagrams are for informative use only though they shall reflect the accompanied UML and schema perfectly.

5.7 Element

A named rectangle representing the most basic part of the XML Schema notation. Each represents an XML "Element" token. Each Element symbol can be elaborated with extra information as shown in the examples below.

This is a mandatory simple element. Note the upper left corner of the rectangle indicates that data is contained in this element.

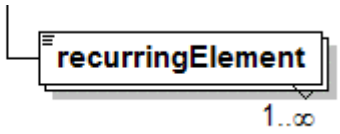
5.8 Optional Element

Optional (non mandatory) elements are specified with dashed lines used to frame the rectangle.

[1] XML Spy: http://www.altova.com

5.9 Recurring Element

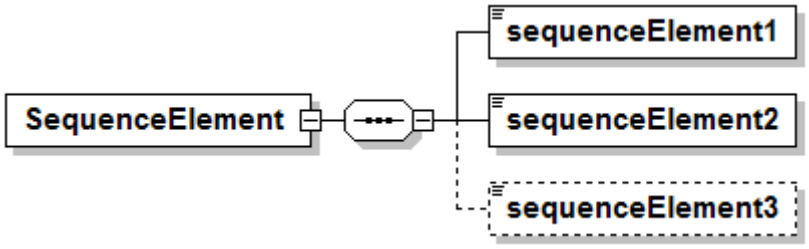
This element (and its child elements if it has any) can occur multiple times.



This example shows a recurring element that must occur at least once but can occur an unlimited amount of times. The upper bound here is shown with the infinity symbol.

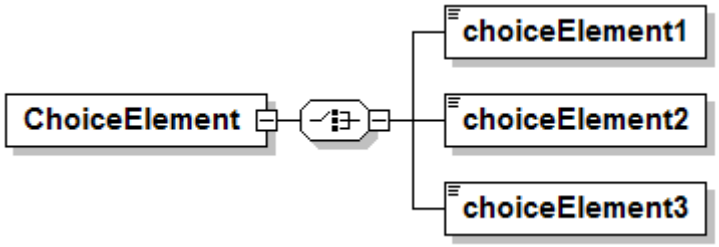
5.10 Sequence Connector

The connection box, called a sequence indicator, indicates that the "SequenceElement" data is made up of three elements. In this example, the first two elements are mandatory and the third element is optional



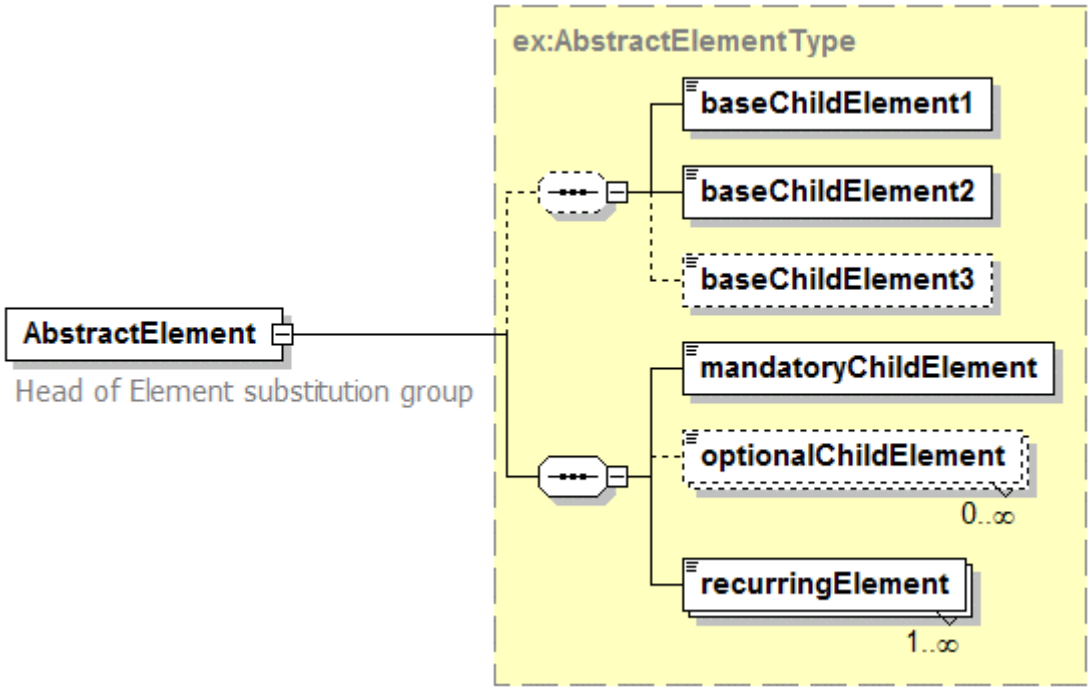
5.11 Choice Connector

The connection box here is a "choice" indicator, indicating that there is always going to be exactly one of the child elements listed on the right.



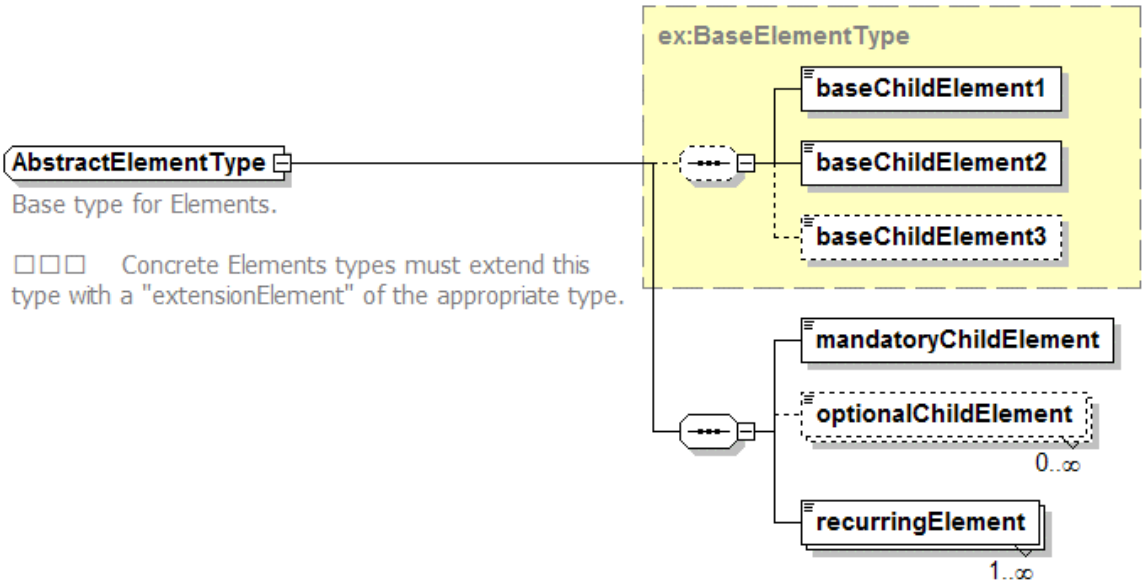
5.12 Definition with Complex Type

This diagram illustrates the use of a complex type (i.e., “ex:AbstractElementType”) for defining an XML element (e.g., “AbstractElement”).



5.13 Complex Type

This diagram illustrates the definition of a complex type (i.e., “AbstractElementType”), extending another complex type (i.e.,“ex:BaseElementType”) with three additional elements. Complex types can be reused to specify that different elements are of the same type.



6 Introduction to TML (Informative)

Transducer Markup Language (TML) is a language for capturing and characterizing not only data from transducers, but information necessary for the processing and understanding of that data by the eventual recipient of the transducer data. Both sensors and transmitters can be captured and characterized within TML, leading to the use of the term "transducer" rather than "sensor". TML handles not only static but also streaming transducer data, and does so in a way which permits the data stream to handle live transducer data both being added to the stream and being deleted from the stream.

Descriptions relevant to the later processing and understanding of the data, including calibration, operational conditions, device settings, transducer properties and characteristics, relationships of transducers to each other in a multi-component system and system behavior are all among the properties captured in a TML description. Logical models, behavioral models, transfer functions, and other information critical to the processing of data are all captured within TML.

TML is capable of precise time-tagging of data, so that it is possible to know precisely when a physical phenomenon was measured at the individual measurement level, and also captures latency or delay information at a fine resolution. This enables the precise determination of when a data point was taken, as well as aiding in interpolation between data points and the reconstruction of events.

6.1 Interoperability and Integration of Transducer Data

The primary purpose of TML is to enable widespread sharing while providing greater understanding of transducer data, such that we can gain a better picture of our world through transducers. Transducers are our interface to the real world. Just as our minds rely on multiple senses to gain a better situational understanding of our environment so must we utilize multiple modalities of transducers to gain a better situational understanding of our world.

To manage the complex task of processing and associating millions of sensory inputs, we must utilize computing machines to coordinate and associate data and represent only interesting data in a summarized manner such that humans can make higher-level decisions in a timely manner. To facilitate this task we must enable a seamless and unambiguous communication language for transducers and computing machines to act as a homogeneous system. To accomplish this a transducer communication language must be complete, consistent, and efficient as possible. Many times these properties are mutually exclusive.

TML is an interoperable solution for the exchange, archive, and common understanding of transducer data

TML is scalable to facilitate a wide variability of application.

6.2 Interoperability and Common understanding of Transducer Data

For interoperability we need:

1) the data which describes the encoding, structure and organization of the transmitted data,
2) the data that describes the structure and organization of the transducer data.

For transducer data understanding we need:

1) the data which describes the transducer response and geometric characteristics,
2) the data which describes any pre processing to the data, and
3) the data that describes external relationships for the transducers and processes within a system.

TML provides a cohesive set of common normalized descriptions to enable the common understanding of transducer data. This common understanding will enable machines to begin to make lower level associations of transducer data (such as conflation or association and registration of features). It is paramount that the data description documents remain closely linked with the data, one cannot be understood without the other.

6.3 Live and Historical Streaming Data

Because transducers are real world interfaces, their responses are representative of the changing world. This changing world is represented in live streaming transducer data. TML captures this living streaming data, which is representative of real world transducer events corresponding to multiple phenomena and maintains the relative and absolute temporal and spatial relationships of the data such that we can review the events exactly as they happened either live in real time or at a different place and/or a different time. TML data represents a continuous stream of data from or for possibly a multitude of different transducers, all interleaved randomly, in roughly chronological order. It is up to the data acquisition equipment to capture the data as precisely and accurately as possible such that the precision and accuracy capabilities of the transducers are fully recognized and the acquisition process is not degrading the data in any fashion. The task of data integration and understanding is left to the transducer processors.

6.4 Variability of Transducers from Simple to Complex

There is a great variability in the complexity of transducers and the applications that process and control them. A transducer system may be composed of several different transducers as well as processes that prepare the data. Transducers as simple as a temperature probe and as complex as an interferometric synthetic aperture radar system must be able to be integrated into a single homogeneous transducer representation of our world. Transducer work in two directions, data not only comes from transducers (e.g. sensors) but also can be sent to a transducer to produce certain predetermined results (e.g. hydraulic actuator). There is currently no single data format that can adequately capture and communicate this information on such a wide range of complexity of transducers, other than TML.

6.5 Variability in uses and applications

Transducer data is used in a variety of application for a variety of uses. For example transducers have widespread use in medical, mineral exploration, environmental, industrial process control, military reconnaissance and surveillance, building security, weather, and air space control, only to mentionto name only a few. There is also great variability in complexity of processing transducer data. TML supports complex as well as simple processing and it is domain or application agnostic. In other words, it does not make any assumption on how the data will be processed, displayed, or used, nor does it contain any domain specific administrative data. TML is ideally suited for sharing transducer data across application domains. Applications may also demand that transducer data be from live sources as well as from archived sources. Live sources of transducer data may be from mobile, portable, or stationary transducers. TML is configured to handle all the variability in uses and applications efficiently.

6.6 Variability in communications

TML complies with the open system philosophy of protocols. TML may be carried by any session level protocol on a network (e.g. FTP, HTTP, XMPP, etc.). This also means that it may be used with any network or transport protocol such as UDP/IP or TCP/IP. TML is scalable such that it works for passing transducer data with the simplest physical connections such as RS-232 over a point-to-point connection. TML does not provide access control, or error detection and correction. These tasks are provided by services. TML is efficient and robust such that it can be implemented in a wide variety of communication architectures. Everywhere from a simple SIMPLEX point-to-point wire or RF connection to Multi-channel, wideband secure networks.

6.7 Variability in Processing

Within a global transducer web architecture there is the potential of for a great variability in the transducers, transducer data, applications, as well as processing capabilities. Processing platforms can range from cell phones and PDA to powerful multi-processor workstations and mainframe computers operating in both central and distributed processing architectures. The less capable PDA will not be able to handle the highly complex data from sophisticated transducers. There must be a procedure for handling this situation. By matching the complexity of the data with the processing capabilities of the processing machine, the user will know immediately if his machine is capable of handling the data or if they should look for some processing help on the network. TML implements a concept of defining a *complexity level* for transducer data. This *complexity level* is based on the computing resources required to process the data in *real-time*. The complexity level is also based on communication resources required for communication the data. See section 6.6 variability in communications.

6.8 Static and Dynamic Data

TML utilizes a programmable data structure that adapts to each individual transducer. It also utilizes a common model or description to describe the various complexities of the transducer data. A primary objective of the TML is to exchange the transducer data as

accurately, and efficiently as possible. This is accomplished by exchanging two types of data

1) the dynamic data which represents the time varying world, and

2) a description of the data such that a TML enabled processor would be able to understand the various complexities of the data for a particular transducer system.

The data description is required to understand the data and the data is required to understand the data descriptions. Because of the very common possibility that data description data is changing it relies on the streaming data to carry the time sensitive parameters as sensor data. It is imperative that a logical connection be kept between these to types of data.

6.9 Normalization of Phenomena

Enabling a wide diversity of transducer data to be made available opens doors to vast capabilities in the understanding of the world. Multiple modalities of sensed data implemented properly and consistently can be synergistically combined to determine much more knowledge about an object or feature than single mode sensing. One key to this fusion of data and understanding is a well thought out phenomenology database. It is paramount that a standard structured normalized database of phenomenon and phenomenon properties be established for the critical task of feature recognition from multiple modalities of sensing.

6.10 TML Generic Architecture Components (a systems view)

This section is informative; its purpose is to give the reader an example implementation so they may understand how TML is used in an operational architecture. *TML* is of no use unless it is implemented in an operational architecture. Figure 7 - Generic TML implementation is an example high-level system view of a generic architecture showing the various components. *TML* data can be exchanged directly (point-to-point) between a *Transducer System* and a *Transducer Processor/Controller*. If the TML transducer node has a network adapter, then *TML* data may also be routed through a network. *TML transducer nodes* may support services such as registry searches of local archives through persistant or stateless network concetions. Figure 7 - Generic TML implementation shows an example implementation with four different adapters to the TML data. The adapters are the transducer, process, network, and application adapters. All of these adapters may utilize many of the same functions. The application adapter will interface the TML stream to the client application, which may be a transducer processor, controller, or both. To the user, data from the *transducer system* is received in *TML* and the transducer interface is transparent to the application. The unique interfaces to the transducers and processes are adapted to a TML interface through a TML transducer and process adapter, respectively. TML Network Nodes provide facilities for managing multiple process and transducer systems on the network. They will enable a client user to choose which transucers and processes they need to accomplish their required task. All *transducer systems* and *process system*s connected to the network should be available to any user on the network (pending security constraints) because TML provides a common

interface. All of the proprietary and unique sensor interfaces are hidden from the user. That difficult task was assigned to the *Transducer System* integrator to ensure the proper *Transducer Adapters* were installed to understand the unique communication with each *transducer* in the *system*.

The TML Application Nodes are a common set of software libraries which form the interface to the TML data. These libraries will support parsing, decoding, workflow chaining and basic TML processing and control and can be used by a wide variety of applications. It will be necessary for TML data to be processed and controlled by a variety of applications. For each application an application adapter will be required such that the application can utilize the functions provided by the TML Application Node.

To take advantage of TML universality there will need to be services on a network that will enable search and discovery of transducer capabilities, transducer system live data feeds, transducer archive data, transducer processing services, security, transducer system planning, and scheduling services.

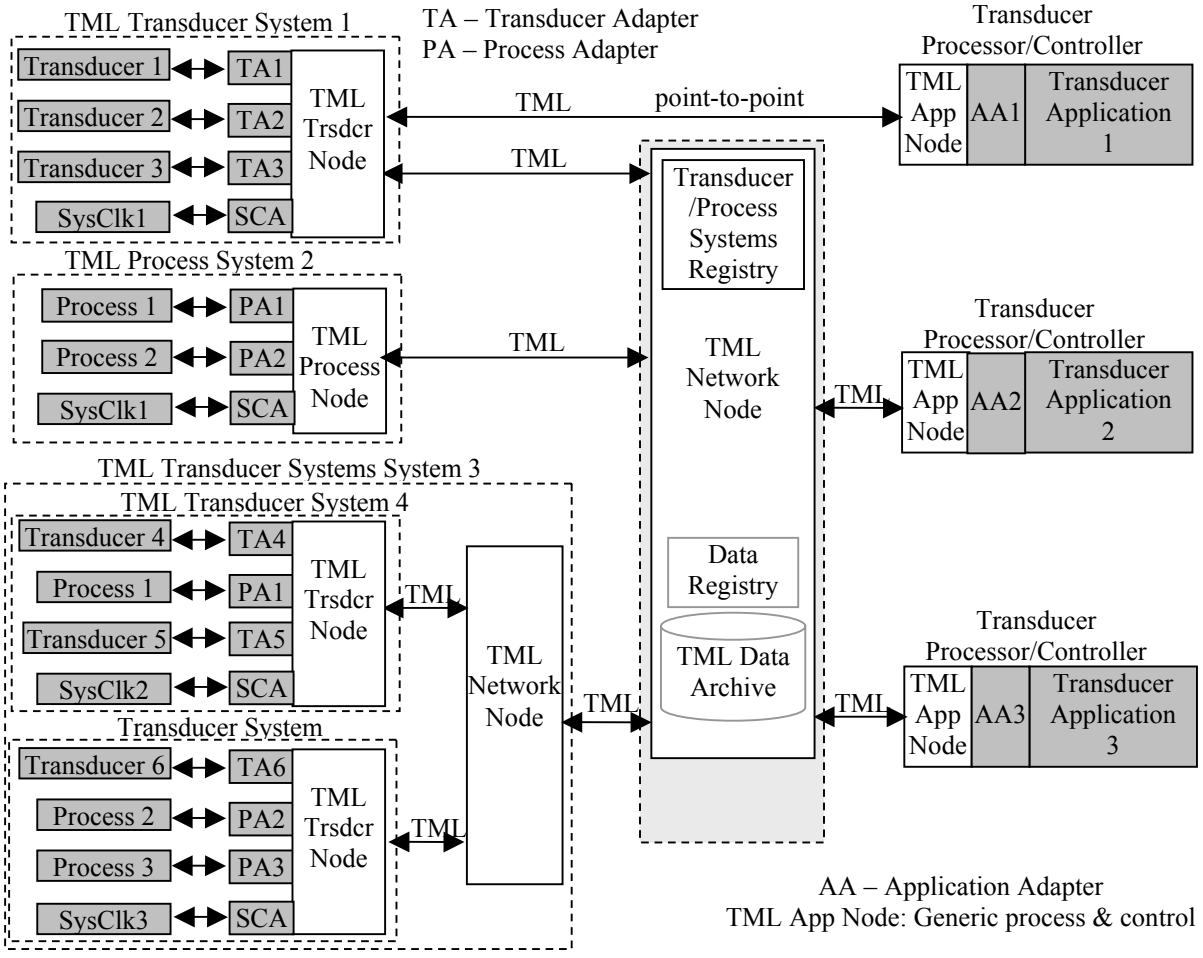


Figure 7 - Generic TML implementation

Registry Services and the Identification of Transducers and Transducer Descriptions, Transducer Processes and Transducer Process Descriptions, Transducer Data, and Transducer Data Descriptions

Fundamental to any distributed network architecture is the ability to find and uniquely locate resources on the network. A transducer network is no exception. Web clients must be able to search on a system registry, transducer registry, and process registry.

6.11 Modular descriptions

TML modularizes the data documentation such thatso that each stakeholder can encapsulate documents without modification. Many people may be involved in the development of the system description documentation and not all these people will develop documentation at the same time. Transducer manufactures will develop a transducer description for their transducer. Transducer processing services will develop documents to describe their processes, and system integrators will develop documents to describe their systems. When building a system one must be able to include the descriptions, without modification, created by the individuals who designed or integrated the sub components that are being put into a system. The user will then add to those descriptions the characteristics of the system as the user has integrated it. For this reason, the user must be able to reuse characterizations done by other people and insert them into the descriptions when using their components.

6.12 TML data stream

The TML *data stream* is a product generated from a *TML system*. The *TML data stream* carries the time varying data representing various external and internal *phenomena*. The opening TML tag initiates a stream. A stream may be promptly terminated at any point at which time the reading machine should add a closing TML tag to make the terminated stream valid XML. If the stream is terminated normally a closing tag from the sender will terminate the stream.

TML implements a time tagged implementation of XML. What this means is that a system time clock count is inserted into the start tag of TML data elements to signify the relative time (from the sysClk) (or absolute time, in some cases) to indicate when the data contained in each TML element was acquired. The system clock should be of sufficient resolution to adequately relate time differences at a transducer sample sub-sampling interval (approximately an order of magnitude faster that the fastest sample clock in the system) and enough digits to minimize the possibility of a roll over.

In the example system below, Figure 8, five transducers (Camera, Image Size, GPS, IMU, & Compass) and two processes (JPEG Compress and Base64 encode) generate five types of data clusters.

Types of Data Clusters from the example shown in Figure 19.

1) Base64 encoded-JPEG Compressed-Digital Video,
2) JPEG Compressed File Size (IMAGE_SIZE),
3) GPS,

4) IMU, and
5) Compass.

Cluster number 1 is from a process. The digital camera generates 15 video frames per second and each video frame has 640 x 480 16-bit pixels. The data rate from the digital camera was too high to capture directly so it was compressed with a JPEG compression process. To put the data in an XML data stream we had to convert the binary data to text with a Base 64 encoding process. The data in Cluster 1 represents digital video data that has been JPEG compressed then Base64 encoded. At this point is just one big text blob. When the processor received this data it will know to un-Base64 encode then un-JPEG compress, in that order to get to the video data. The JPEG compression process generates varying size of files depending on the entropy of the image data. This process will generate a value indicating the size of the frame. We chose to handle this value as a separate sensor reading. We could have just as well incorporated it into the Base64-JPEG-Video cluster as two data Units, a length value, and a binary blob.

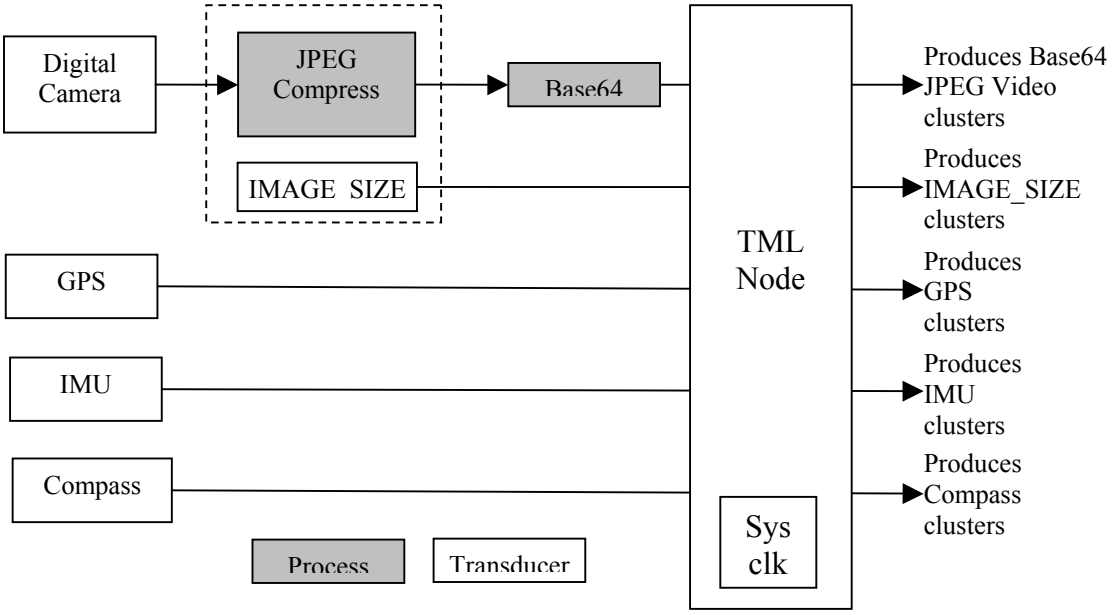


Figure 8 - Example TML system

The TML data stream contains the changing values from the individual transducers. The following example XML data is an example of streaming data from the example system shown above. Each <data> element contains a *data* cluster from the respective transducer or process. The ref attribute in the <data> start tag contains a reference to the description of the data contained in the element. In the following example, the five sensors described above are streamed in chronological order. The following stream described the state of the *transducer system* over time. In some situations, data may come out of time order. TML processors should be aware of this and buffer data long enough to account for this. Figure 9 shows how TML data could be exchanged over a network.


```
<data clk='28118774' ref='IMU'>22.8,1.1,3.4</data>
                              <!—IMU: true heading, pitch roll-->
<data clk='28118792' ref='COMPASS'>21.1</data>
                              <!—COMPASS: mag heading-->
<data clk='28118795' ref='GPS'>516866,-4702126,4264297.2005-08-
26T16:31:49Z</data>
                              <!--GPS: X, Y, Z, Time-->
<data clk='28118800' ref='IMAGE_SIZE'>49094</data>
<data clk='28118800' ref='CAM'>...base64 JPEG video...</data>
<data clk='28118874' ref='IMU'>23.9,2.7,-1.1</data>
<data clk='28118888' ref='Weather'>0,15.3,35,18.8,18,0.0,22.5,
82.3</data>
   <!--WX: rain,dewPt,humid,temp,wndchl,wndSpd,wndDir,baroPres-->
<data clk='28118899' ref='IMAGE_SIZE'>49388</data>
<data clk='28118899' ref='CAM'>... base64 JPEG video...</data>
<data clk='28118974' ref='IMU'>0.1 -1.2 1.1</data>
<data clk='28118999' ref='IMAGE_SIZE'>49252</data>
<data clk='28118999' ref='CAM'>... base64 JPEG video...</data>
<data clk='28119174' ref='IMU'>-0.1 1.3 -0.2</data>
<data clk='28119199' ref='IMAGE_SIZE'>49628</data>
<data clk='28119199' ref='CAM'>... base64 JPEG video...</data>
<data clk='28119227' ref='COMPASS'>22.5 0.2 -1.2</data>
```

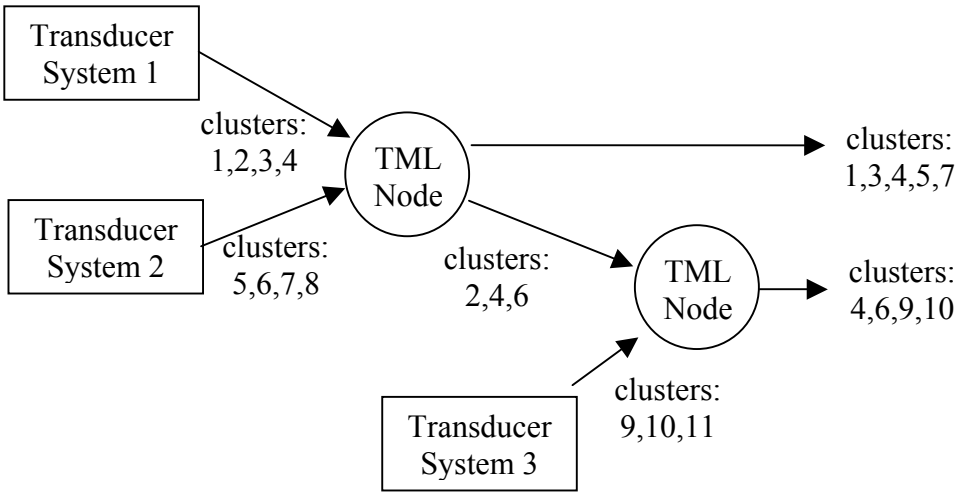


Figure 9 - Merging and splitting of data clusters through network nodes

As data from transducer systems is published on the internet, it can be split and merged to meet individual user needs. TML network nodes will manage the splitting and merging of clusters and maintain the data description documents such that they adequately reflect the data. As new systems come online there will be a need to update the data description. The TML stream may, from time to time contain updates to the customized data description of split and/or merged streams.

6.13 Asynchronous sampling

A system may employ multiple transducers to observe or take action on multiple phenomena at the same time. Because not all phenomenon have the same natural frequency, the different transducers will sample at different rates determined by the natural frequency of the phenomenon or other factors. To understand what the state of the system of a transducer is at any point in time it is important to time tag the data so that data can be interpolated between updates.

6.14 Sequence and timing of clusters

TML data streams interleaved clusters from multiple transducers onto a serial channel. In many multi-transducer systems, the individual transducers are not all triggered to capture their data at the same instance in time. The transducers will trigger at different rates. Figure 10 illustrates a system of four sensor sampling at different rates. To establish what the state of the four transducers (sensors) are at any instant in time their values will have to be interpolated between samples. This can only be accomplished if the times of each sample have been carefully time tagged with a high-resolution clock. It is important that a single clock be used to maintain relative timing relations between all the sensors samples.

The sequence of the clusters may be skewed slightly because of data buffering issues. Time tags in the cluster start tags facilitate in maintaining relative timing relationships of the data clusters in the data stream. Using the time tag will enable the clusters to be properly sequenced at the destination by ordering data clusters by time.

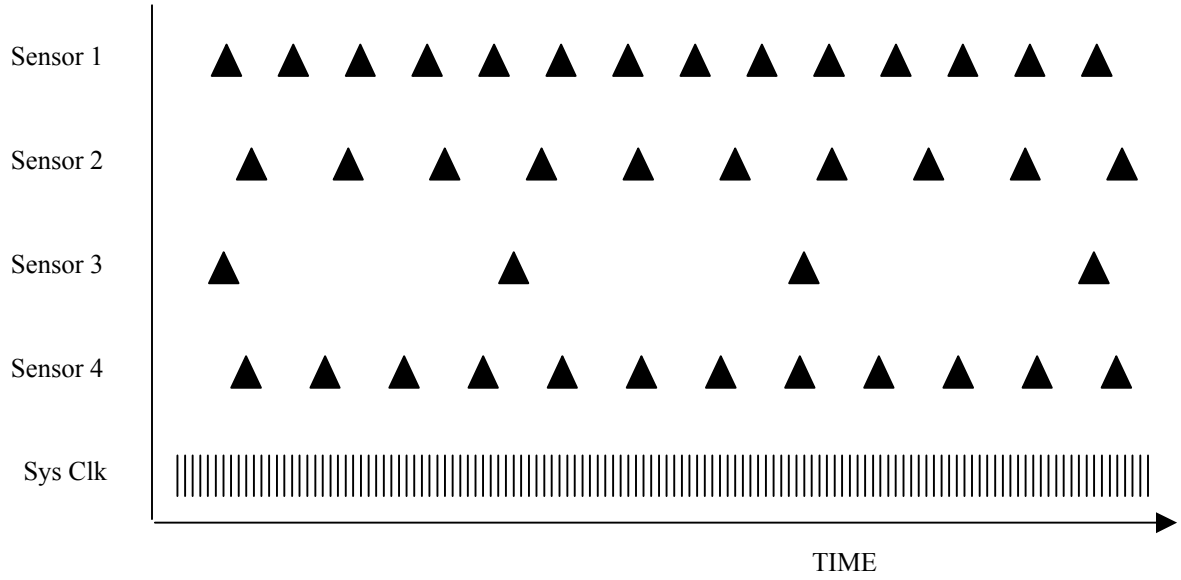


Figure 10 - Unsynchronized transducer triggering

6.15 Verbose and non-verbose time and reference attributes

The start tag of each data element contains a “clk” and “ref” attribute or a “dateTime” and “reference” attribute. These are relative versus absolute time and reference attributes of the data cluster with the data tag. The “clk” attribute contains the state of the system clock at the first instance of data in the data cluster. The “ref” attribute contains the relative address, which points to the data description for the description of data contained in the data cluster. If there exists a need to send the data in a self-contained package. , Then then the full verbose versions of time and reference ID (dateTime and reference) attributes may be used in the start tag. The shortened relative version or terse version is to minimize data overhead in the exchange and archive process.

6.16 Transducer Characteristic Frame (TCF)

The TML data frame, known as a Transducer Characteristic Frame (TCF), is a concept and model of the information which is encoded and documented through the transducer data model, data cluster description, and transducer behaviour models. The TML data frame enables the behaviour models to be associated with the transducer data.

The definition of a TCF is dependent on characteristics of particular transducers and systems, and is independent of any XML encoding. Data can be thought of as being in groups made up of samples, which are in turn made up of measurements. TML allows the definition of TCFs to suit particular applications to attain a full or characteristic reading of phenomena.

TML data clusters can be transformed into any XML encoding that accurately expresses the above groupings. It may be viewed as a transport mechanism independent of SWE client encodings and primarily concerned with the fusing of data through temporal and spatial correlation.

A TCF has the following structure:

- The meaurement is an individual value at the smallest level
- A sample can contain one or more measurements
- A TCF contains one or more samples
- Each data cluster contains one or more TCFs

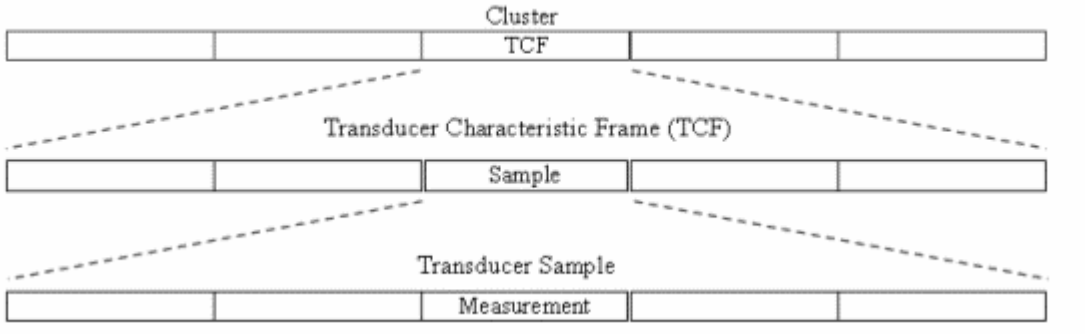


Figure 11 - TCF

Figure 11 above illustrates the TCF. The TML data model corresponds to one TCF, and the definition of a TCF is based on what is appropriate for a particular transducer. For example, a temperature sensor making single measurements might have as its TCF a grouping of measurements, while in the case of a camera, an individual pixel would constitute a measurement while a line within the image may be a sample, and the entire image would be its TCF.

> Fundamentally a TCF can be thought of as an array of n-cells. The number (n) of cells is determined by the number of unique spatial and temporal coordinate values required to characterize one elemental period of the transducer. TCF's are classified by dimensionality. The TCF is a key concept for modeling the internal relationships of a transducer. The TCF enables response and geometry model data to be associated with the transducer data. Transducer data will be captured such that the boundaries of the TCF are resolvable. There will be a one-to-one correlation between the data TCFs and the model TCFs. The modeling and the data are closely related and tight relationship should be maintained, if they are not exchanged together. The number of spatial coordinates assigned to each data unit determines the dimensionality of a TCF.

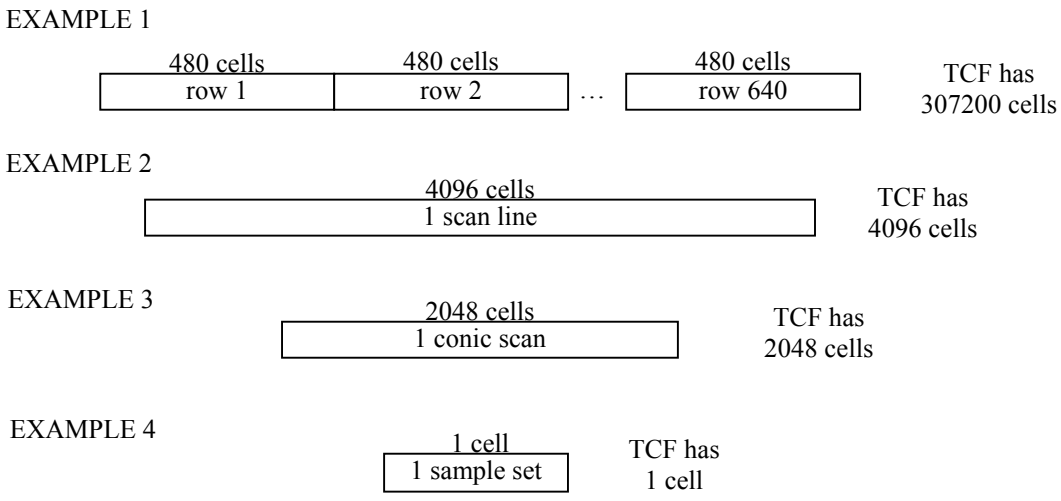


Figure 12 - TCF examples

EXAMPLE 1: a TCF for a camera representing one frame with an array of 640 x 480 has 307200 cells in the TCF, 640 columns and 480 rows. The dimensionality of the TCF is 2. There are two spatial coordinates assigned to each cell or data unit in the array, an α coordinate, and a β coordinate.

EXAMPLE 2: a TCF for a single detector line scan sensor representing one line, with an array of 4096 x1 has 4096 cells in the TCF, 1 row and 4096 columns. The dimensionality of the TCF is 2. There are two spatial coordinates assigned to each cell or data unit in the array, an α coordinate, and a β coordinate.

EXAMPLE 3: a TCF for a conical scan LIDAR sensor representing one conical scan, with an array of 2048 x 1 has 2048 cell in the TCF. The dimensionality of the TCF is 2. There are two spatial coordinates assigned to each cell or data unit in the array, an α coordinate, and a β coordinate.

EXAMPLE 4: a TCF for a positioning system which measures its own x,y, & z position relative to a reference, produces a measurement every second, with the x coordinate, y coordinate, and z coordinate of its position. This sensor has a TCF with only one cell. There is no spatial significance to its position with the array, and there are no spatial coordinates assigned to describe interior geometric orientation, therefore this sensor has a dimensionality of zero (0).

NOTE: All in situ transducers will have a TCF dimensionality of zero (0) because they have no ambiguity space.

6.17 Transducer order re-sequencing for transport.

As transducer data is prepared for transport, it must be serialized. In the serialization process, data may be reorganized or re sequenced from the original transducer data structure. For example, a four-detector line scan sensor, which produces four lines by 1K pixels long on each swath or spin of the mirror, has a TCF of 4K pixels or 4K dataSets. The order the pixels are generated are the firstas follows: First pixels in each of the four lines are created at the same instant then the second pixels are created in the next instant. Corresponding pixels in the four lines are all sampled in parallel. The transducer data structure would be to sequence the pixels in the first line then the pixels in the second line, then the third, then the forth. Transporting the pixels may or may not conform to the spatial organization of the pixels. The transport may favor the temporal order of the pixels where it would send the first pixels of each row before sending the second. As you can see, there is multiple ways of transporting data. The objective is to be able to put it into the transducer order (spatial order). TML does not prefer one method to the other. Whichever method is chosen, then the sequence of the pixels or data units in the TCF will be described by a sequencing TCF which is described in the data product description element. The sequence TCF is a serial count. If the count started at 1 and increased by 1 until the number of data Sets in a TCF is reached then the transport sequence would be the same as the transducer (spatial) order. Refer to the data index definition in the terms and definition section to see how index numbers are sequenced to determine the transducer order. There are short cuts that can also be taken such that not every sequence number needs to be listed.

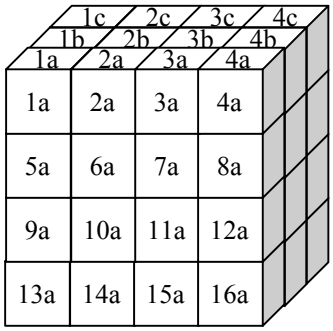


Figure 13 - Index numbers for describing sequence pattern

Transporting TCF data can be done in many different ways. The sequence element describes the order in which data units are transported.

Example 1 sequence

1a,2a,3a,4a,5a,6a,7a,8a,9a,10a,11a,12a,13a,14a,15a,16a,1b,2b,3b,4b,5b,6b,7b,8b,9b,10b,
11b,12b,13b,14b,15b,16b,1c,2c,3c,4c,5c,6c,7c,8c,9c,10c,11c,12c,13c,14c,15c,16c

Example 2 sequence

1a,1b,1c,2a,2b,2c,3a,3b,3c,4a,4b,4c,5a,5b,5c,6a,6b,6c,7a,7b,7c,8a,8b,8c,9a,9b,9c,10a,10b
,10c,11a,11b,11c,12a,12b,12c,13a,13b,13c,14a,14b,14c,15a,15b,15c,16a,16b,16c

Example 3 sequence

1a,5a,9a,13a,2a,6a,10a,14a,3a,7a,11a,15a,4a,8a,12a,16a,1b,5b,9b,13b,2b,6b,10b,14b,3b,7
b,11b,15b,4b,8b,12b,16b,1c,5c,9c,13c,2c,6c,10c,14c,3c,7c,11c,15c,4c,8c,12c,16c

Example 4 sequence

1a,1b,1c,5a,5b,5c,9a,9b,9c,13a,13b,13c,2a,2b,2c,6a,6b,6c,10a,10b,10c,14a,14b,14c,3a,3b
,3c,7a,7b,7c,11a,11b,11c,15a,15b,15c,4a,4b,4c,8a,8b,8c,12a,12b,12c,16a,16b,16c

Example 5 sequence

1a,2a,3a,4a,9a,10a,11a,12a,5a,6a,7a,8a,13a,14a,15a,16a,1b,2b,3b,4b,9b,10b,11b,12b,5b,6
b,7b,8b,13b,14b,15b,16b,1c,2c,3c,4c,9c,10c,11c,12c,5c,6c,7c,8c,13c,14c,15c,16c

Example 6 sequence

1a,1b,1c,2a,2b,2c,3a,3b,3c,4a,4b,4c,9a,9b,9c,10a,10b,10c,11a,11b,11c,12a,12b,12c,5a,5b
,5c,6a,6b,6c,7a,7b,7c,8a,8b,8c,13a,13b,13c,14a,14b,14c,15a,15b,15c,16a,16b,16c

Example 7 sequence

1a,2a,3a,4a,8a,7a,6a,5a,9a,10a,11a,12a,16a,15a,14a,13a,1b,2b,3b,4b,8b,7b,6b,5b,9b,10b,
11b,12b,16b,15b,14b,13b,1c,2c,3c,4c,8c,7c,6c,5c,9c,10c,11c,12c,16c,15c,14c,13c

Example 8 sequence

1a,5a,9a,13a,14a,10a,6a,2a,3a,7a,11a,15a,16a,12a,8a,4a,1b,5b,9b,13b,14b,10b,6b,2b,3b,7b,11b,15b,16b,12b,8b,4b,1c,5c,9c,13c,14c,10c,6c,2c,3c,7c,11c,15c,16c,12c,8c,4c

Example 9 sequence

4a,3a,2a,1a,8a,7a,6a,5a,12a,11a,10a,9a,16a,15a,14a,13a,4b,3b,2b,1b,8b,7b,6b,5b,12b,11b,10b,9b,16b,15b,14b,13b,4c,3c,2c,1c,8c,7c,6c,5c,12c,11c,10c,9c,16c,15c,14c,13c

Example 10 sequence

4a,4b,4c,3a,3b,3c,2a,2b,2c,1a,1b,1c,8a,8b,8c,7a,7b,7c,6a,6b,6c,5a,5b,5c,12a,12b,12c,11a,11b,11c,10a,10b,10c,9a,9b,9c,16a,16b,16c,15a,15b,15c,14a,14b,14c,13a,13b,13c

6.18 TML and binary data

Because XML cannot efficiently carry binary data, as discussed previously, TML enables the binary data to be sent or exchanged via binary means. The method to send or exchange binary data will be described in future versions of this specification.

6.19 Other data products

The TML data descriptions are designed to describe TML data. However, there is some utility in using TML data descriptions to describe other common forms of legacy data product formats such as Tagged Image File Format (TIFF).

6.20 Functional system description

The transducer descriptions provide basic descriptions not only of transducer data but how that data is produced and transducer behavior. The data cluster description provides the parameters to enable interoperable exchange of transducer data. The various parts of data description documentation (transducer behavior, process behavior, data description and relations between various components of the system) provide the two parts necessary for interoperability and data understanding.

The data description contains detailed descriptions of the functional components that comprise a transducer system. The fundamental components are: a system clock, transducers, processes, and relationships. Each of these will be described in more detail.

6.21 Common system clock

TML data philosophy requires that when acquiring data from multiple sources the data should be time tagged with a common clock. Using multiple clocks makes it more difficult to correlate data to high precision if required. Figure 10 in section 6.13 illustrates how a common system clock can time tag transducer sampling events so that relative and absolute time relations can be maintained throughout the exchange and archive process. It is possible to provide a precise time stamp for every data unit within a data stream with very little overhead. With proper time stamping and temporal

calibration of the transducer the instantaneous state of all transducers can be calculated for any instant in time.

6.22 Transducer components

A transducer reference model is shown in **Figure 41**. This is a very simple model. The transducer is basically the interface between the data world of computers and the real world. All information and artificial actions go through a transducer. A transducer therefore transforms a real world phenomenon into data or vice-versa. TML will characterize the What, Where, and When aspects of transducer data. The following sections will discuss this in more detail.

6.22.1 Time, space, and value aspect of transducer data

6.22.1.1 Temporal modulation

For computers to handle data from transducers the data must be digitized. Since all transducers are analog devices, their analog signals are converted to a digital signal through an analog to digital sampling process. The frequency of this sampling process is dependent on the highest rate of change (of the analog signal. The analog signal is a direct function of the phenomenon signal. To accurately capture the state of phenomenon, the transducer must be able to respond as fast as the phenomenon changes (or is modulated), and the analog to digital sampling process must sample the analog signal at a rate to adequately represent the analog signal in digital form. If analog data is sampled at least at twice its highest frequency the digital data will adequately represent the analog signal in most cases. The rate of data is therefore dependent on the rate at which phenomenon can be modulated. TML characterizes these frequencies and timing considerations of transducers, so that it is known when the limits of the transducer have been exceeded and the data should not be trusted.

6.22.1.2 Spatial modulation

Phenomenon can also be spatially modulated. In other words, the state of a property can vary as a function of location, such as the optical reflectance of the earth's surface. From outer space there is a different reflectance value for every point on the earth. The reflectance value of the optical phenomenon is spatially modulated. Imaging sensors are a class of transducers that detect spatial modulations as well as temporal modulations of phenomenon. Just as the temporal modulation has frequency consideration for transducers so do spatial modulations of phenomenon. There are types of phenomenon where the spatial modulation frequency is too high to be detected by even the best imaging sensors. Transducer spatial characteristics such as modulation transfer functions are required to be known if the resolution of an image is important. There are many forms of imaging transducers. Spatial and temporal modulation transformations of the transducer must be understood of the transducer before we can accurately model the real world.

6.22.1.3 Value

The accuracy of a transducer response is dependent on the transducer's ability to represent the space and time modulated phenomenon. Besides being able to track the frequencies of change, the transducer must also be able to accurately represent the magnitude of the phenomenon. When the temperature goes up one degree the transducer must not report something different. The transducer must also accurately represent the relative or absolute value of the phenomenon.

To understand the data from a transducer, or to understand what data to give a transducer we must understand the space, time and value aspects of the transducer (or the what, when and where questions).

6.22.2 Transducer modeling concept

Ttranducer modeling data is necessary to provide a TML application with the necessary information to process TML data from a TML transducer or process system. The transducer characteristic frame is utilized extensively in the modeling of transducers and in defining how transducer data should be interpreted.

6.22.2.1 Relating model data to the transducer data using the Transducer Characteristic Frame

TML uses the Transducer Characteristic Frame to correlate the data to the data descriptions. Transducer data is captured in data clusters and the data cluster is a mechanism for improving the transport efficiency of communicating the TCF over the communication channel. A cluster will optimally contain one TCF. However if the TCF is very large or extremely small then the larger TCFs can be split into multiple clusters, or the smaller TCFs can be grouped into a larger cluster. Whatever the situation the TCF boundaries must be readily identifiable.

There are many parameters which are used to describe transducer systems which may be variable by their cell or data index location within the TCF array. Geometry and response characteristics may vary as a function of the cell location within the TCF. For example, the interior geometric modelling parameters are a type of data description that will utilize the TCF for containing modelling parameters. The ambiguity shape has a shape and position relative to a spatial reference system. For a case of an imaging camera, this is comparable to saying that a pixel has an instantaneous field of view (IFOV) and each IFOV for each pixel in an image frame has a unique angular offset relative to the imaging sensor reference frame.

Each cell or indexed data location within the TCF of the geometry model will have a unique coordinate which corresponds to the corresponding cell or data index location in the TCF used for sending the data in the data cluster. A modelling TCF is typically constant for a transducer system. In some cases the internal modelling may change. In these instances a virtual sensor will be tasked to monitor the changing parameters. Figure 14 illustrates the previous discussion. In this figure the modelling data indicates the relative spatial position and relative timing of each of the pixel samples in the ten sample

per line linescan sensor. Each scan starts with the first pixel being sampled at the clock time. The first pixel corresponds to -1.0 unit's alpha angle and a 0.0 unit's beta angle relative to a transducer reference system. The next pixel is 0.2 time units after the time tag value, an alpha angle of -0.7 units and a beta angle of 0.0 units. The scan sequences through the ten samples or pixels until it reaches the last sample in the line. The last sample is sampled at 1.0 time units after the time tag and is at a spatial coordinate of 1.0 alpha angle units and 0.0 beta angle units relative to the transducer reference system.

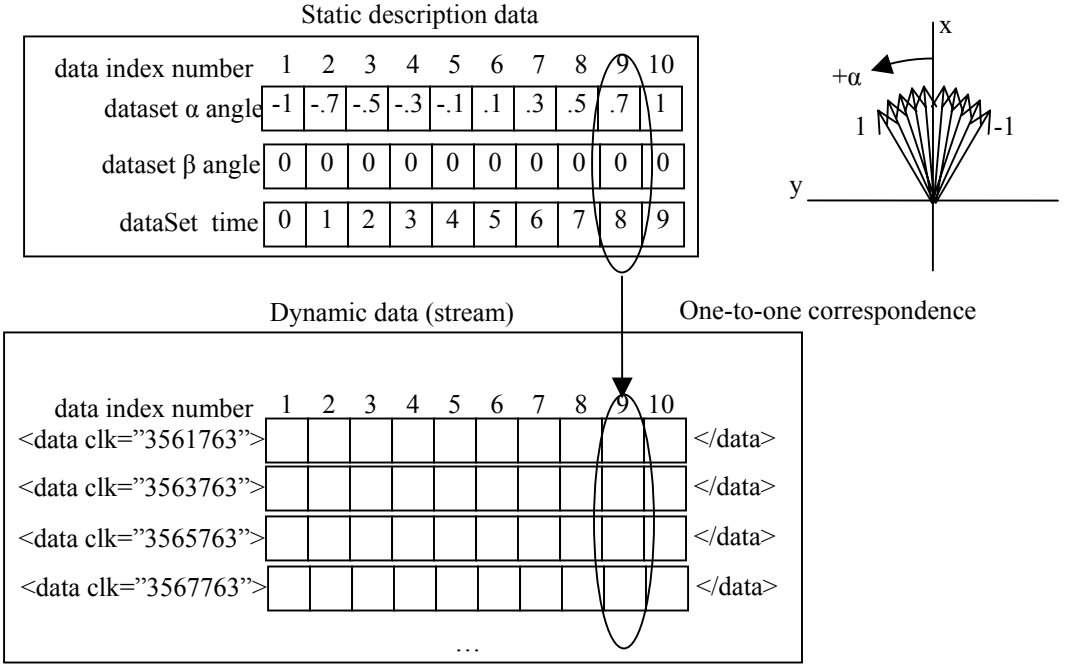


Figure 14 - Using the TCF to relate modelling data to streaming data

6.22.3 Transducer Data Structure

The transducer data structure is comprised of three levels. The most basic data component is the dataUnit. One or more dataUnits shall compose a dataSet and one or more dataSets shall compose a Transducer Characteristic Frame (TCF). Figure 15 illustrates the hierarchy of the transducer data structure. To put it into perspective, this structure can be related to an image. The TCF is the image frame, the dataSet is the pixel, and the dataUnits are the red, green, and blue components. Other types of transducers may not have as complicated of a structure. For example, a spectral radiometer will have a TCF of one dataSet, and a dataSet of several dataUnits. An even simpler structure is an audio microphone. The TCF contains one dataSet, and a dataSet contains one dataUnit.

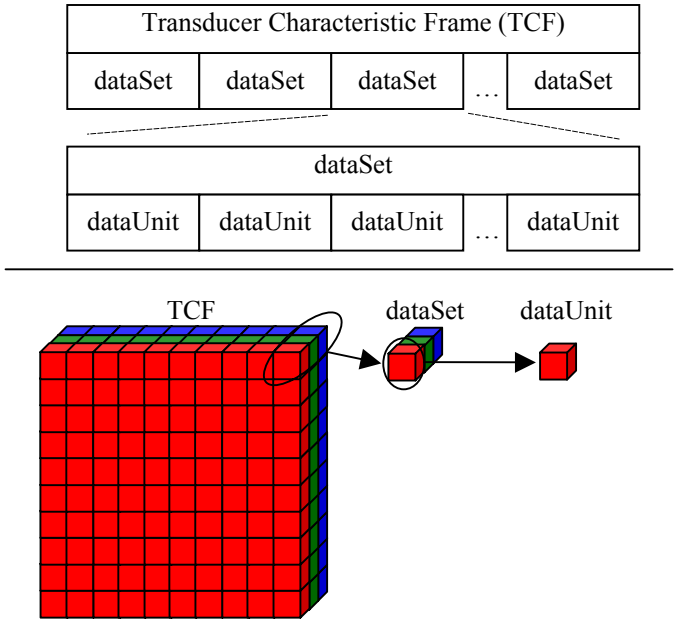


Figure 15 - Transducer data structure

The TCF when exchanged is a serial chain of data. However, it may have a well-organized data structure of columns, rows, and/or planes. If the data structure is well-organized then, the transducer data structure will be organized giving priority to the spatial distribution of the TCF cells. The temporal distribution will affect the direction through the sequence and may affect transducer data structure when there is no differentiation in the spatial structure (e.g. radar). In Figure 16 the transducer data structure for the line scan sensor is a single line of ten datasets comprising one TCF. The TCF begins with the rightmost scan position at time zero and sequences through to the leftmost scan position at nine time units after. Each cell in the TCF is given a data index number, corresponding to its transducer data structure order. When this TCF is transported or exchanged, it may be transported out of sequence. If the sequence of the data index number is sequenced exactly as the data is re-sequenced then the transported data structure can be re-sequenced back into the transducer data structure.

One TCF of 10 dataSets

data index number	1	2	3	4	5	6	7	8	9	10
dataset α angle	-1	-.7	-.5	-.3	-.1	.1	.3	.5	.7	1
dataset β angle	0	0	0	0	0	0	0	0	0	0
dataSet time	0	1	2	3	4	5	6	7	8	9

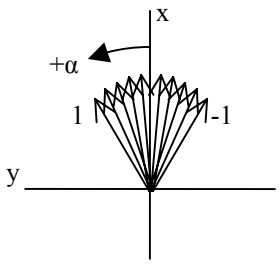


Figure 16 - Data structure for a simple linescan sensor

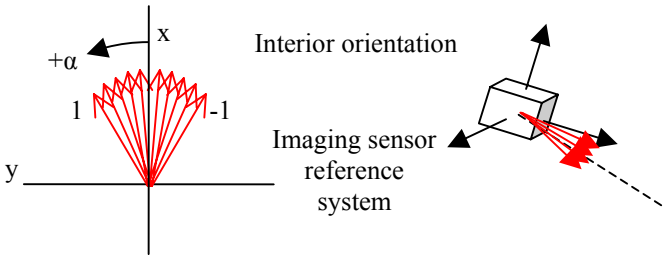
It should be noted that the sequencing of the structure in the transport may be different than the sequence of the structure as it is defined in the transducer model.

DataSets shall contain one or more dataUnits. DataUnits within a dataSet shall share similar spatial and temporal coordinates or characteristics. There may be slight internal differences among dataUnits within a dataSet which may be described through offsets. However dataUnits sampled at different rates or dataUnits with very different spatial coordinates shall not be mixed into a single dataset.

6.22.4 TML Geometry Model

The geometry model describes the mapping of data to real world spatial and temporal coordinates or ambiguities whichever the case may be. The real world geometry relations of data are described using interior or relative geometry, and exterior or absolute geometry.

Interior geometry can be described such that it is independent of the location, attitude, date, and time. Once the interior geometry is defined then it can be positioned relative to absolute spatial and temporal data. Figure 17 shows the interior geometry of a simple line scan sensor. The interior geometry is described with the aid of the TCF.



One two dimensional TCF of 10 dataSets

data index number	1	2	3	4	5	6	7	8	9	10
dataset α angle	-1	-.7	-.5	-.3	-.1	.1	.3	.5	.7	1
dataset β angle	0	0	0	0	0	0	0	0	0	0
dataSet time	0	1	2	3	4	5	6	7	8	9

Figure 17 - Interior geometry relative to transducer reference system

Figure 18 illustrates the exterior geometry of the line scan sensor. The interior geometry is positioned (location and attitude) relative to an external absolute reference system (in this case the earth reference system).

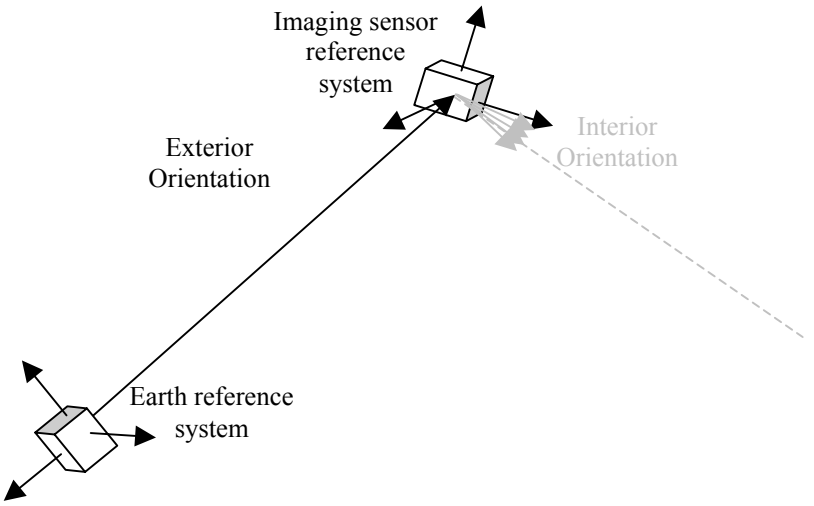


Figure 18 - External geometry relative to absolute reference system

Transducers may be stationary, portable, or mobile. In the case of stationary transducers, their position can be measured and recorded and it will remain the same as long as it does not move. Portable sensors are much the same, they can be surveyed in position, and their position reading will remain unchanged. For mobile transducers the external geometry may not be determined directly in many cases as in the case of the stationary and portable transducers. The location and attitude of mobile transducers may be measured by several transducers working together. As in the case shown in Figure 19, the imaging sensor is positioned using a chain of location and attitude measuring sensors. To determine the external position of the imaging sensor, at any instant in time, the state of each of the sensors responsible for calculating its position (of the imaging sensor) must be determined. This is a reason for applying precise time tags to sensor data.

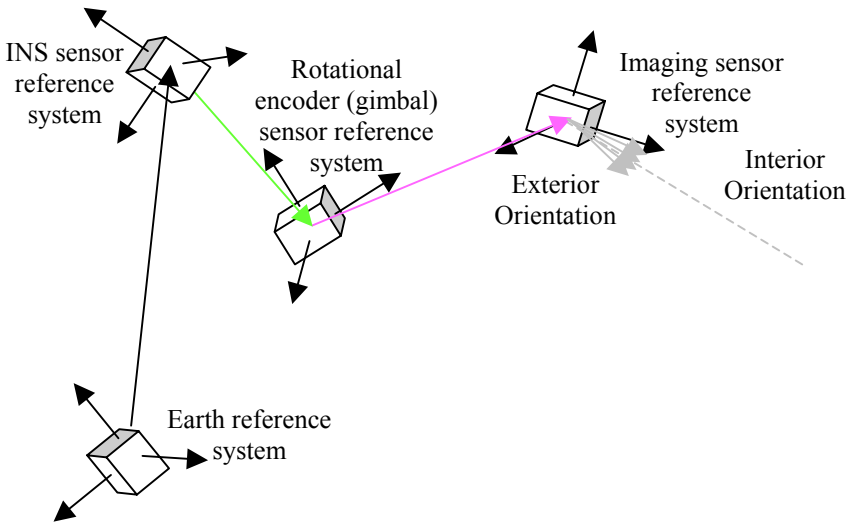


Figure 19 - External geometry through several transformations

In-situ transducers sample their data at the site of the transducer. There is nothing that can be determined of the space other than at the site of the in-situ transducer unless some external knowledge is known of the sensing medium and its phenomenon propagation characteristics. Since in-situ transducers only apply to a single location and there is no field associated with the transducer, in-situ transducers do not have an internal geometry. Only the external is required to describe the position and orientation of the in-situ transducer. In-situ transducers have no ambiguity space, but in some situations in-situ transducer data may be able to determine information about a location other than the transducer location by knowing details and properties of the feature or medium where the transducer is located. An in-situ transducer may be either *attitude or location invariant* as well.

EXAMPLES: in situ receivers - compass, temperature probe, displacement or velocity sensors; in-situ transmitters – stepper motor, heater, mode switch.

In contrast to in-situ transducers, a transducer which creates a phenomenon at a remote location or detects a phenomenon from a remote location is known as a remote transducer. The remote location is a location other than where the physical transducer is located. A remote transducer may have an ambiguity space associated with it, or it may apply or detect a phenomenon to a specific location (no spatial ambiguity).

Transducer data geometric characteristics are factors of both spatial and temporal characteristics. Both spatial and temporal characteristics have relative and absolute considerations. The following paragraphs will discuss these relationships.

6.22.4.1 Spatial Characteristics

To determine where transducer data is (its position) spatial models of transducers are used. Both relative and absolute positioning are used to determine where transducer data is. Although the absolute position of transducer data is of the most concern, to get there it is necessary to first understand the relative positioning and then transform the relative position into an absolute position with the aid of other transducers.

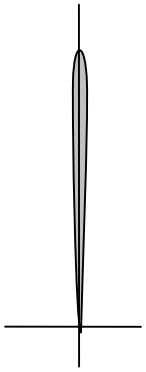
6.22.4.2 Ambiguity Space

To improve the sensitivity or improve the effective power density or improve spatial resolution, a focused field of energy or received area may be described. This is ambiguity space. The ambiguity space is a space where it is highly probable that the object or feature creating the phenomenon is located. For example, an Instantaneous Field of View of a camera pixel is an ambiguity space for that camera. The antenna pattern for a radar is an ambiguity space for that radar. There are energy field ambiguities refereed to in TML as Instantaneous Field of Influence (IFOI). Transducer data may have multiple ambiguity spaces associated with it. For example after radar data is received and it is associated with the transmit pulse, each dataUnit in the received range gate has a unique range associated with it. It is therefore possible to determine that objects within the transmit and receive antenna pattern and at a constant range (sphere) could contribute to the response.

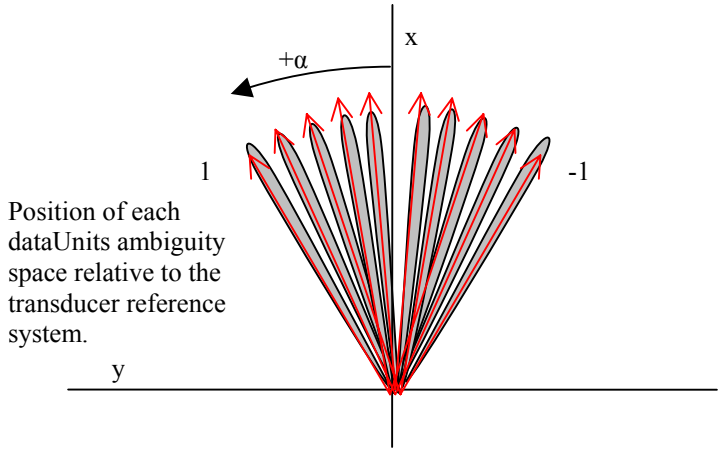
The ambiguity shape may take on a variety of shapes. The shapes may be well formed or they may be irregularly shaped such as a terrain model. Ambiguity shapes may be described using any of the coordinates available in the three coordinates systems available in TML (rectangular, spherical, and cylindrical) relative to any of the reference systems (ECEF, Local, Transducer).

6.22.4.3 Internal spatial characteristics

The internal spatial characteristics of transducer data are described by characterizing the ambiguity shape and position for each data unit. Typically, the ambiguity shape will be basically the same for every data unit in the TCF. There may be small variances over the array, which can be described by an array variable. However, the position of the ambiguity shape for each data unit is different. Figure 20 illustrates the characterization of the internal geometry relative to the transducer reference frame and the start of the scan. In this case, the spatial position of the ambiguity shape is described relative to a transducer reference system and the temporal reference is described relative to the start of the scan. The alpha and beta angles define the orientation of each of the ambiguity shapes relative to the transducer reference system. If the reference system is defined properly the positional variances of each of the data units may change only one or two coordinate parameters out of the six possible (three location and three attitude). For example, in this case the scan was in the direction of constant beta. Only one coordinate varied, alpha.



Ambiguity shape of a single dataUnit



One TCF of 10 dataSets

data index number	1	2	3	4	5	6	7	8	9	10
dataset α angle	-1	-.7	-.5	-.3	-.1	.1	.3	.5	.7	1
dataset β angle	0	0	0	0	0	0	0	0	0	0
dataSet time	0	1	2	3	4	5	6	7	8	9

Figure 20 - Distribution of ambiguity shapes relative to a transducer reference frame

A transducer reference system is the only engineering reference system (see OGC abstract publication Topic 2) allowed for use in referencing transducer data. The transducer reference systems axis are defined with respect to what makes it easiest to describe the spatial distribution of the transducer data. The transducer reference system is chosen to accurately represent the position of the ambiguity space for each data unit within the TCF data Array. The transducer reference system should correspond to the transducer reference system described in the physical system description. The physical reference system description will indicate where the origin of the axis is in relation to the transducer device and the orientation of the principle axis. In some cases, the transducer spatial reference system may move relative to the physical reference system. In this situation, sensors are required to monitor its movement relative to the physical reference. Objects in the transducer spatial reference frame can be positioned with either a rectangular or a spherical coordinate system.

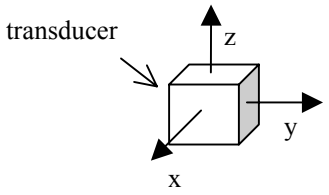


Figure 21 - Transducer spatial reference system

EXAMPLE the transducer reference system for a camera may be chosen such that the origin of the coordinate axis is located at the frontal node of the optics and the x axis points along the focal axis of the lens, with the y axis pointing out the right side of the camera aligned with the horizontal scan of the camera raster. Such that each column in the focal plane array will have a common α angle, and each row in the array will have a common β angle. Giving each pixel in the focal plane a unique α and β spherical coordinate relative to this transducer reference system.

NOTE: For most transducers the position and orientation (attitude) of the transducer reference system is important. In some instances, only the position or orientation of the reference frame is important. See location and attitude invariant transducers

6.22.4.4 In-situ transducers do not have an interior spatial characteristics

A transducer may sample (sensor) or create (actuator) a phenomenon that is determined to exist only at the site of the transducer (in situ) or the phenomena may be the result of propagated energy, which is affected by a feature or an object from a remote location (remote transducer). Even from remote transducers, which have precise internal and external geometric calibration, unless the propagation medium and the feature of interest characteristics are understood, geometric associations are not possible. Measurements from an in situ sensor or actions by an in situ actuator cannot be applied to any other location around the in situ transducers.

6.22.4.5 External spatial characteristics

The external geometry positions (location and attitude) relative reference systems (typically the transducer reference system) give the relationship to an absolute reference system as well as placing an absolute time on the relative internal time. The external orientation of transducers may be either static (as in stationary and portable transducers) or dynamic (as in mobile transducers). The external geometry is typically measured with transducers. Therefore, transducers measure positions of transducers, creating a multi transducer system. Within these multiple transducers, it is necessary to understand the relative positioning of each of the transducers to one another such that when the position of one of them is known the position of the other can be determined. These transducers in the multiple transducer system will invariably sample positioning data at varying rates and accuracies. This requires the user to be aware of the way data is time tagged and measurement uncertainties are characterized. Figure 22 shows a multi-transducer system where the GPS and IMU are capturing the state of the external orientation of the line scan sensor. The GPS and IMU sample location and attitude data at different rates so they are time tagged such that the state of the system can be determined at any instant.

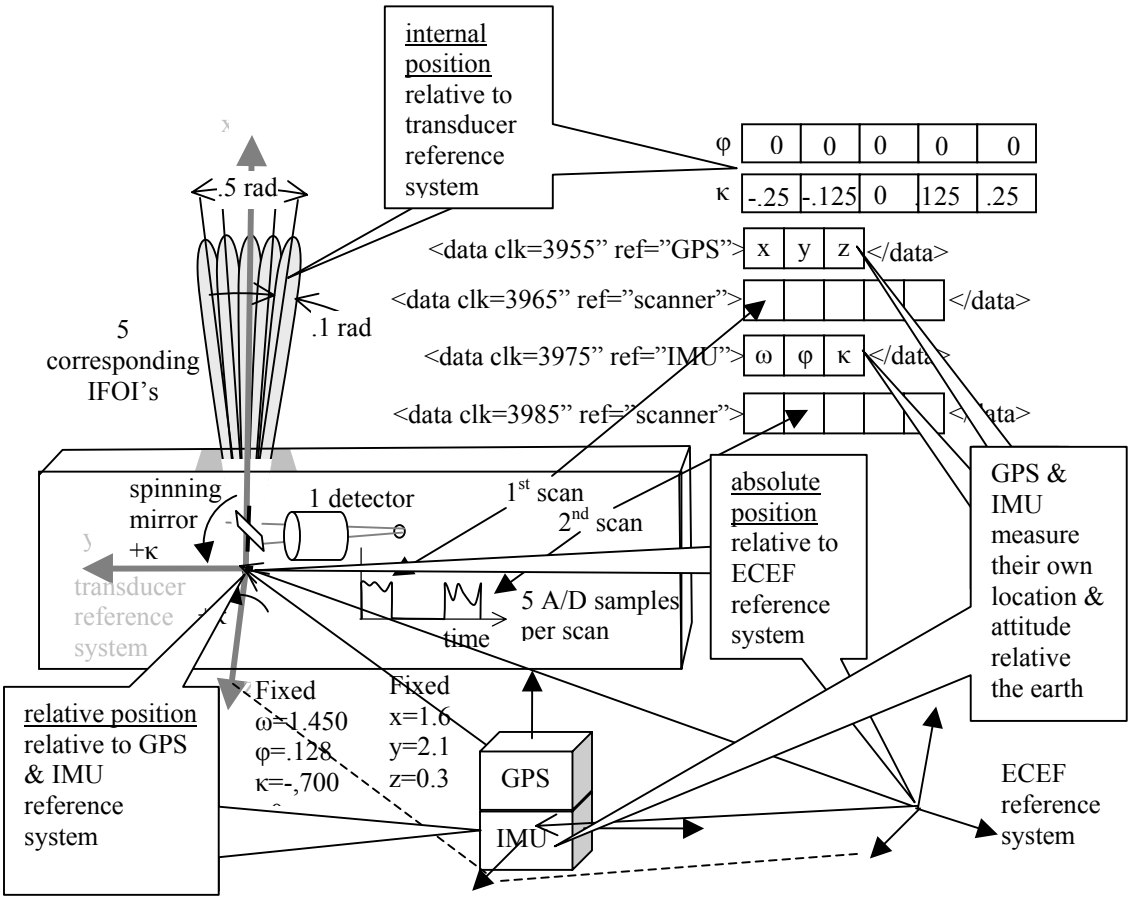


Figure 22 – Internal (relative) and external (absolute) position

6.22.4.6 Temporal Characteristics

Just as important as determining where data is, is determining when data is captured. In many cases, the time of data must be determined to determine where the data is. The temporal and spatial characteristics work together to describe an overall geometric characterization of a transducer system. The TCF is utilized to aid in the description of the internal temporal characteristics for a transducer. The internal temporal characteristics also utilize the relative time reference of the TCF trigger or the time of the first data in the TCF. The time of the first data is t=0.

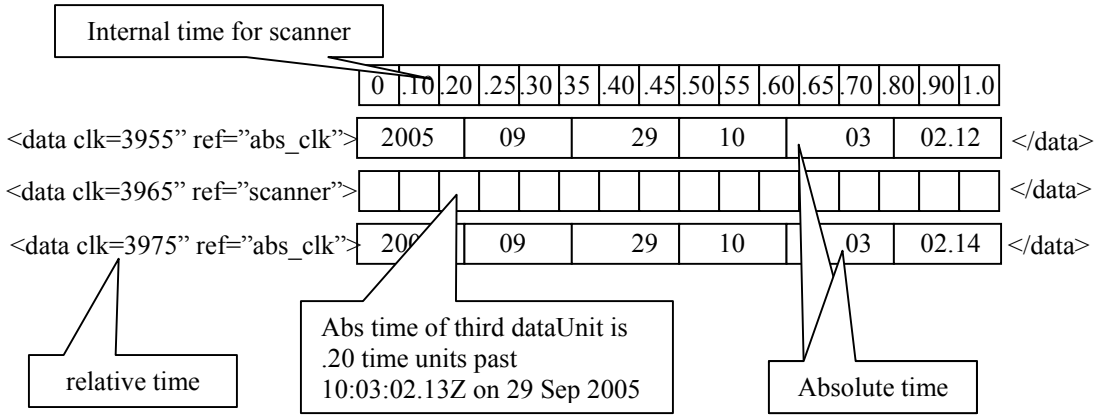


Figure 23 - Internal (relative) and external (absolute) time

6.22.5 Other Geometry models

Hooks have been put into TML to incorporate other geometry models such as Replacement Sensor Model (RSM), Rational Polynomial Coefficient (RPC), and Tactical Sensor Models (TSM). The incorporation of these models is intended to support the interim transition of legacy systems still producing data using the sensor models. For interoperability and fusion considerations, it is recommended that the use of these models be limited in TML.

6.22.6 TML Response Model

For calibrated receivers (sensors) the response model is used to determine what the phenomenon was based on the result of the data. For transmitters the response model is used to determine what the action or phenomenon is going to be based on the data sent to it. Factors that influence the response model are integration time, latency time, noise figure, frequency response, impulse response, and steady state response.

6.22.6.1 Integration time

The integration time is the time required by a receiver to accumulate a reading or the time for a transmitter to apply an action. For example, when a camera shutter opens for 1/30 of a second that is the integration time for that set of measurements. When an aircraft flies, an air sampler opens the sampling instrument at 3pm and closes it at 5pm, then 2

hours is the integration time for that sensor. Any modulation that occurs within the integration time cannot be determined. One value corresponds to response or stimulus depending on if the transducer is a receiver or transmitter respectively.

6.22.6.2 Latency time

The latency time is the time between the actual sampling (sensor) or action (actuator) event and the time the data is available. The time tag should represent time as close as possible to the actual time of an event, however there may be a delay until data is actually available. Latencies can range from microseconds to days.

6.22.6.3 Noise figure

Noise is an inherent property of all transformations. The noise figure is a measure of merit for how much noise is added by a transformation of data to phenomenon or vice-versa.

6.22.6.4 Frequency Response

The frequency response is a characterization of how a transducer responds to the various frequency components of the stimulus. TML will use the response amplitude versus frequency for three types of frequency related functions: carrier frequency response, modulated frequency response, and power spectral density. The example in Figure 24 shows how a low pass response of a particular transducer would apply to the carrier or modulated response. When the phenomenon contains frequencies above the cut off point, the response is not able to track it and it will be averaged.

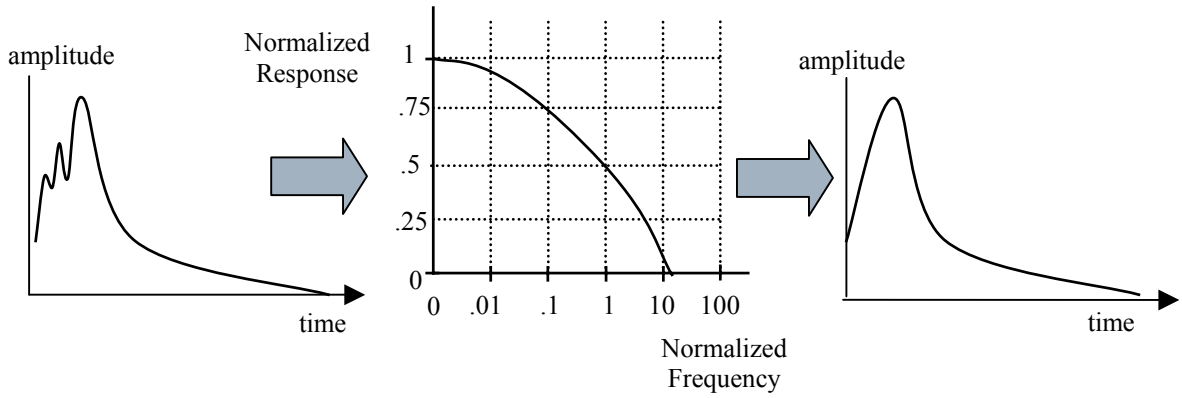


Figure 24 - Low pass response of a transducer

6.22.6.5 Impulse response

The impulse response is useful for characterizing linear time invariant (LTI) transducers. If the transducer's response in known for a unit impulse input, then the transducer's response can be derived by convolving the input signal with the unit impulse response.

6.22.6.6 Steady state transfer function

The steady state transfer function provides a mapping of the phenomenon to the data for the steady state. This function does not characterize dynamic characteristics of the transducer. This function does however characterize linear and non-linear regions of the transducer. To illustrate the steady state function Figure 25 shows a transducer model that senses the temperature property of the thermal phenomenon.

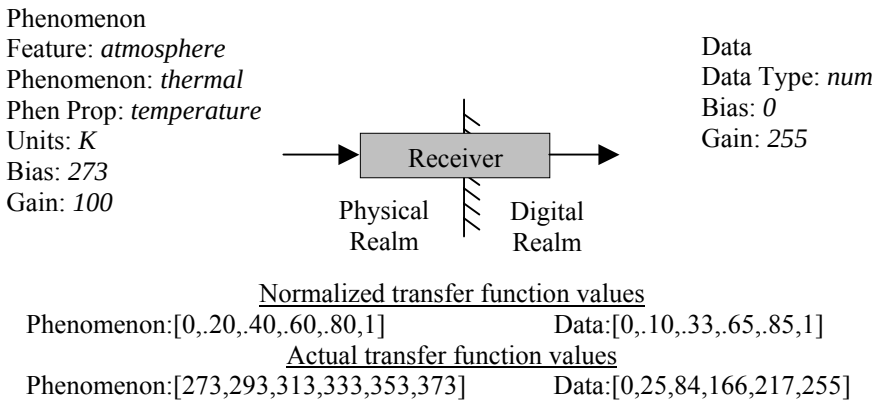


Figure 25 - Model of temperature transducer

A normalized steady state transfer function describes the mapping from phenomenon to data. The normalized coordinates given in the model must first be multiplied and offset the proper amount to represent the actual transfer function. In this example, the normalized phenomenon values were multiplied by 100 then offset by 273 to derive the actual phenomenon scale. The normalized data was multiplied by 255 with no offset to derive the actual data scale. Figure 26 shows this process.

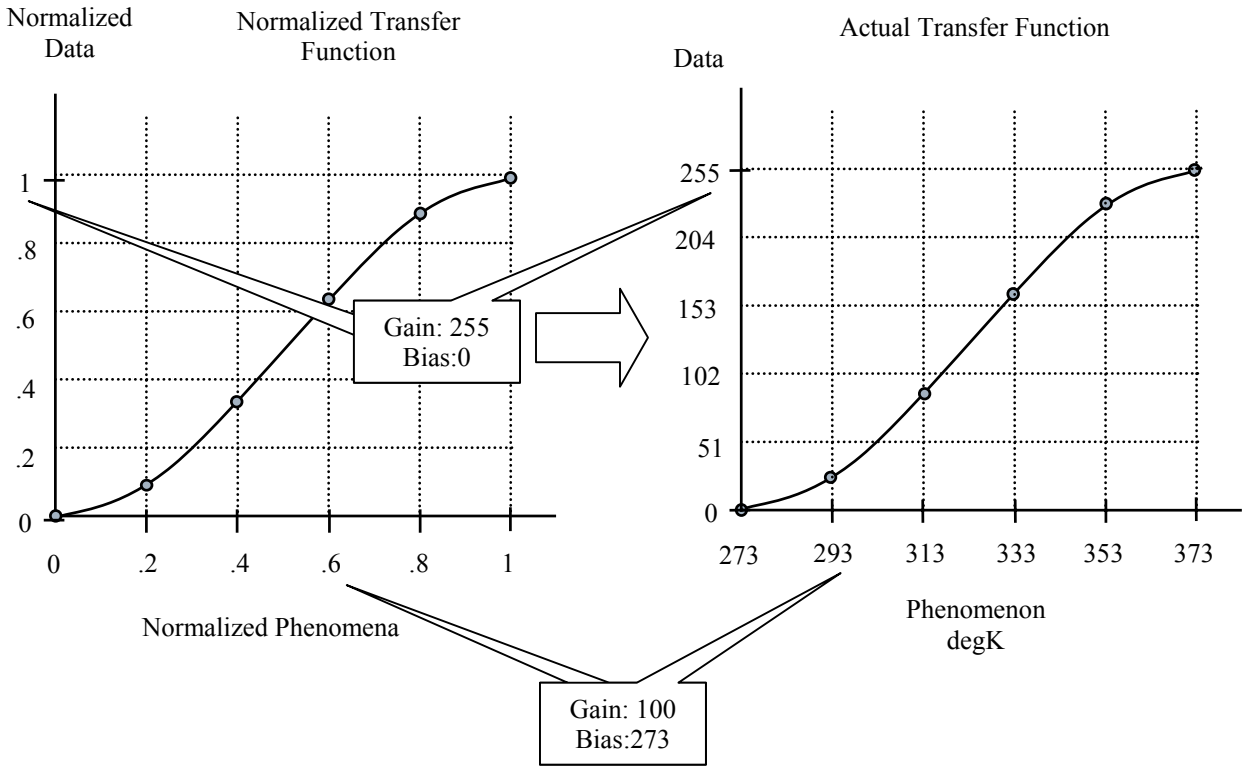


Figure 26 - Deriving the actual transfer function from a normalized function

Some transfer functions use data other than numbers. There are cases where the values represent quantities other than numeric values. The quantities may correlate to a single phenomenon value or a range of values. Examples of non-numeric quantities are Empty, Full, Small, Large, Left, Right, High, Low… The following is an example of a transfer function listing using non-numeric data values.

Normalized transfer function values

Phenomenon:[0-.40,.41-.80,.81-1] Data:[“low”,”medium”,”high”]

Phenomenon Gain: 100, Phenomenon Bias: 273

Actual transfer function values

Phenomenon:[273-313,314-353,354-373] Data:[“low”,”medium”,”high”]

Event sensors are sensors that listen for a specific event to happen or a threshold of a phenomenon to be exceeded; they will then output a data value. These special event sensors have a non-linear transfer function. Figure 27 is an example of a transfer function from a sensor which responds to an over threshold event. In this example whenever the normalized phenomenon value gets above .4 the data output goes high, and when the value drops below .4 the value goes low. Note that the switching point is different when the phenomenon is increasing than when it is decreasing, this is known as hysteresis.

Hysteresis is a property of a steady state transfer function where the function is different depending on the direction data is moving in. Data may move from low to high or from high to low. If hysteresis is present, the transfer function curve is different in the forward direction (low to high) than it is in the reverse direction (high to low). Depending on the situation, hysteresis may or may not be favorable. For example if a fan motor is programmed to come on when the temperature gets to 50 degrees, then when the fan kicks on the cooling action of the fan immediately cools the air and shuts off the fan, and thus when the fan shuts off the air is immediately warmed and the fan motor turns on again. The rapid cycling of the fan motor would burn it out quickly with all the stopping and starting. If hysteresis was built into the system, then the fan would turn on when the temperature went above 50 degrees, and turn off when the temperature dropped below 40 degrees. The transfer function would indicate that the fan would switch on at 50 degrees, going from low to high and it would switch off at 40 degrees when going from high to low. This system would exhibit hysteresis and it would stop the fan motor from cycling so frequently.

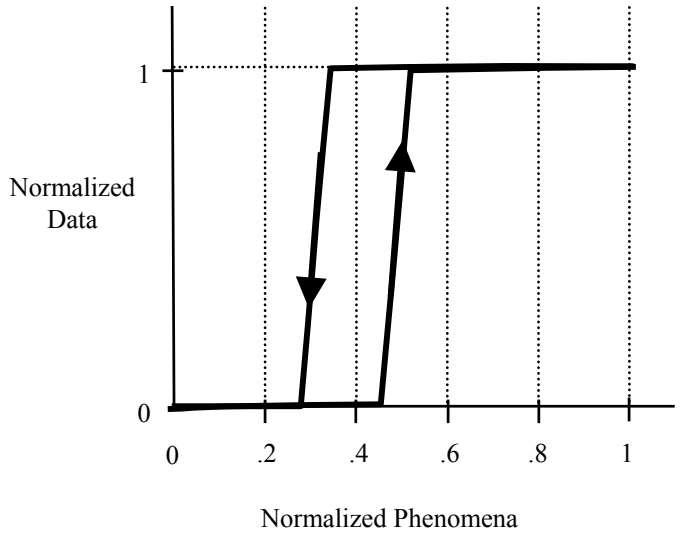


Figure 27 - Transfer function showing hysteresis

The transfer function only shows the range of phenomena that the data range covers. In some instances, the transfer function is dynamic. The transducer may adjust its gain or attenuation of the phenomenon to stay within the dynamic range of the transducer. In these situations, the gain and/or bias factors shall be tracked with a sensing transducer to provide the transfer function scale factors at any instant in time.

A transducer may have more than one steady state transfer function. The transfer function may be different depending on the direction the phenomenon values are going (increasing or decreasing). This hysteresis is characterized separately in each direction.

Figure 28 shows a simple transfer function over ten time units.

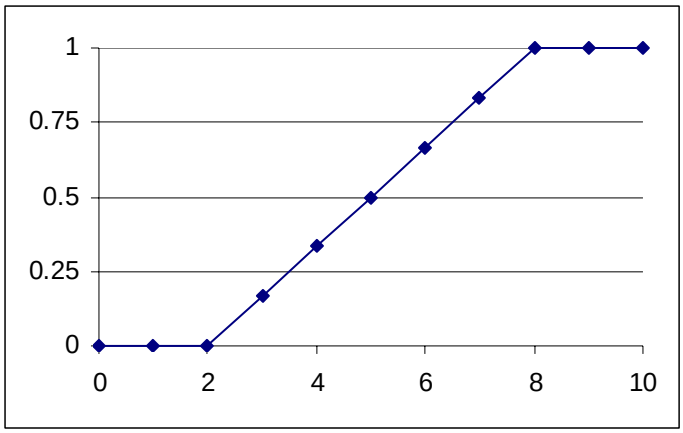


Figure 28 - Simple Transfer Function

6.22.6.7 Interpolation of phenomena state in between transducer events

TML captures transducer events which can occur periodically or randomly. These events represent the state of the phenomena at specific points in time. To determine what a transducer state is in-between events it is necessary to know something about the behavior of the phenomena and the transducer itself. TML categorizes this behavior as either continuous, last state, return to zero or a special RTZ/lastState. Each of the three behaviors is described below.

6.22.6.7.1 Last state interpolation

The last state interpolation is used when the data to or from a transducer is only sent when a transducer value changes (sensor) or is required to change (actuator). This requires less bandwidth for communication.

```
<intTime>0</intTime>

<data clk="00">3</data>
<data clk="11">2</data>
<data clk="39">1</data>
<data clk="44">3</data>
<data clk="52">4</data>
```

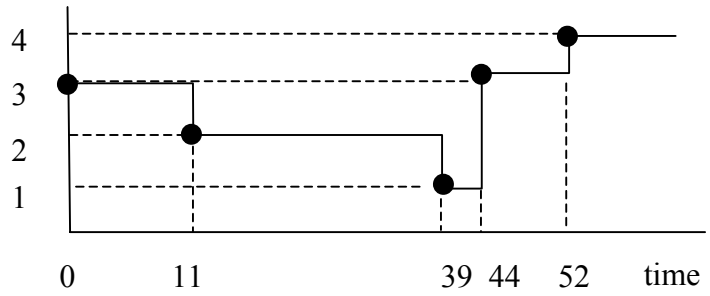



Figure 29 - State (random) events

6.22.6.7.2 Return to zero interpolation

The return to zero interpolation indicates that the transducer is only measuring or acting on something during the TCF of the transducer event. It can be assumed that the phenomenon is zero in between events. The return to zero may work with either periodic or random transducer events.

```
<intTime>2</intTime>

<data t=09>3</data>
<data t=29>4</data>
<data t=39>2</data>
```

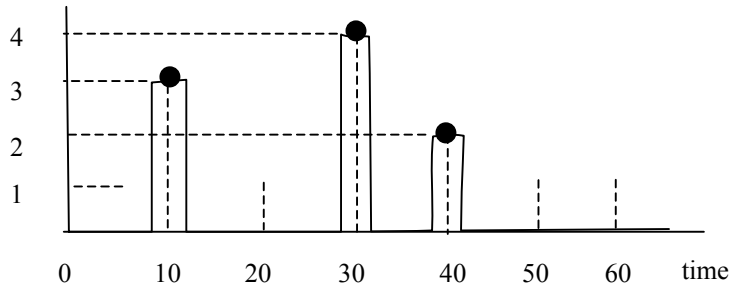


Figure 30 - Return to zero events

6.22.6.7.3 Continuous interpolation

Data interpolation described as continuous means that it can be assumed that data follows a smooth continuous path in between transducer events. Data in between events may be interpolated with any number of interpolation algorithms, depending on the accuracy desired.

```
<intTime>0</intTime>

<data clk="00">3</data>
<data clk="10">3</data>
<data clk="20">2</data>
<data clk="30">2</data>
<data clk="40">1</data>
```



```
<data clk="50">3</data>
<data clk="60">4</data>
```

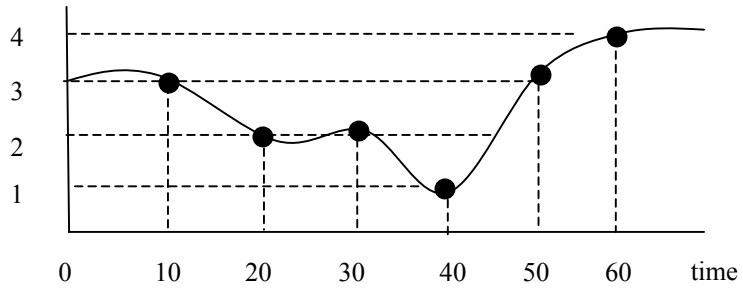


Figure 31 - Continuous events

6.22.7 Miscellaneous

6.22.7.1 Transducer events and triggering

An analog to digital conversion of an analog phenomenon into a digital data representation of that phenomenon (or vice-versa) is a transducer event. Transducer events can occur at millions of times a second or transducer events can happen once a century. The rate at which transducers sample or trigger is dependent on many factors. Implied clock state sensors measure the state of the system clock at the TCF trigger (and if needed the dataset trigger) The clock state sensors value is used in the start tag of the data cluster which contains the data sampled from that trigger event. Figure 32 is an illustration showing the notional concept of the sysClk, triggers, and clock state.

If a TCF from a transducer is sent and the TCF of the data is not it must be assumed that the data contained in the TCF is the same as the last data TCF received. For example, a radar transmit chirp waveform could be captured with its time tag on a transmit pulse of a radar. If every transmit waveform from that point on in the stream is the same, then only the clock state of the trigger for that transmit waveform will be in the data stream. It will be assumed that the data from the waveform cluster describing the transmit waveform was in that spot of the clock state.

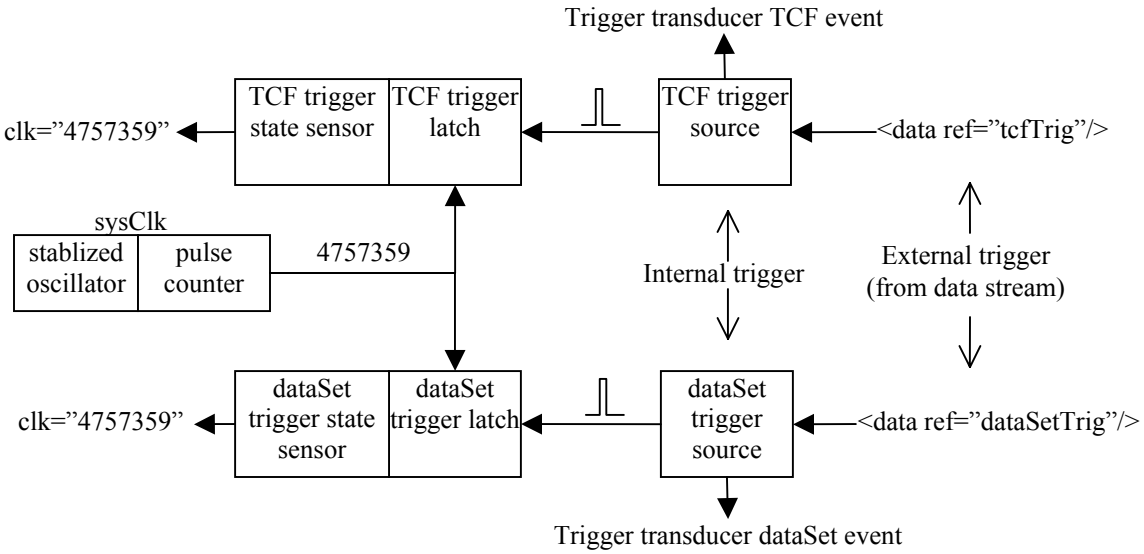


Figure 32 - Transducer internal and external triggering and latching clock states

6.22.7.1.1 Continuous internal triggering

Many transducers sample data continuously, with their analog to digital conversion circuits continually being triggered to take a digital sample of the analog state of the phenomenon. These internal triggering circuits typically trigger the A/D circuits at a periodic rate. This rate is dependent on the frequency content of the analog signal as discussed above. For every sample a transducer makes it outputs a digital piece of data (dataUnit).

6.22.7.1.2 Random internal triggering

Some transducers are a little different, instead varying on random internal triggering. Instead of sending data on every A/D trigger, these transducers only send data when data changes or when data reaches predetermined thresholds. This type of triggering is used with state interpolation. As long as the data does not change, no data is sent. When the data changes it is sent.

6.22.7.1.3 Remote triggering

Other transducers will not invoke an action or take a sample until instructed to do so by an external trigger. The TML system and transducer descriptions enable the description of remotely triggered transducers. When a data element is sent in the data stream that corresponds to the triggering actuator, the transducer system is instructed to trigger the appropriate transducer.

6.22.7.2 Precision and accuracy

In the process of measuring and sampling, errors and uncertainty always play a large role. Every measurement is only so good and some measurements are more accurate than

others. Almost as important as the measurement itself is the uncertainty of the measurement. Every measurement should be qualified with an uncertainty. The uncertainty is characterized by relative and absolute errors from random and systematic sources, and TML provides places to capture accuracy of measurements

6.22.7.3 Function modeling

Functions are used throughout TML to describe things such as frequency response and transfer functions. The function coordinates are described normalized. To convert the normalized coordinates to actual coordinates they must be associated with gain (multiplier) and bias (offset). For example a transfer function that maps phenomenon to data may have normalized phenomenon function values of [0,.20,.40,.60,.80,1] and normalized data function values of [0,.10,.33,.65,.85,1]. If the phenomenon had a gain of 100 and a bias of 273, and the data had a gain of 255 and a bias of zero then the actual phenomenon function coordinates would be [273,293,313,333,353,373], and the actual data coordinates would be [0,25,84,166,217,255]. Figure 33 illustrates an example of how the normalized coordinates with the associated gain and bias factors are used to derive the actual function.

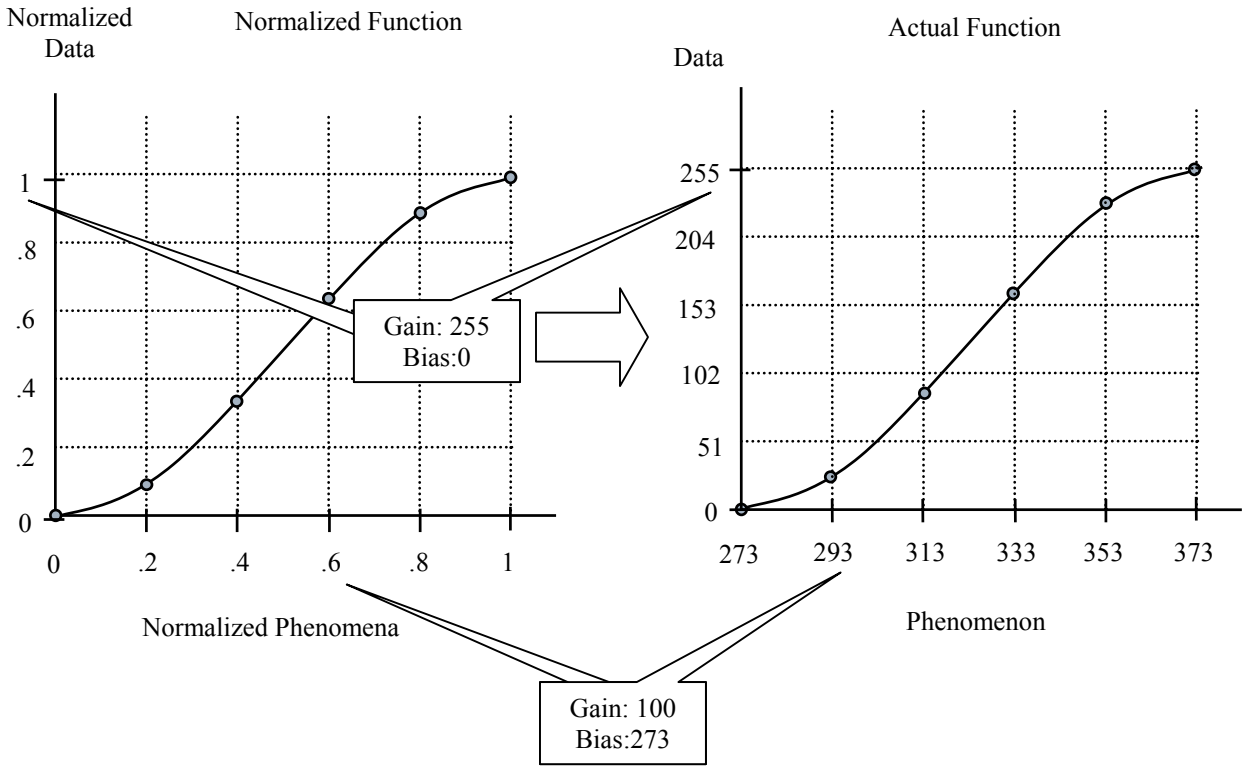


Figure 33 - Actual function derived from normalized function

The function may be a dynamic function where it is changing as a function of time. If this is the case the coordinates themselves or the gain and bias parameters may be read and replaced by sensor readings from the data stream.

6.22.7.4 Ambiguity shape modeling

Ambiguity shapes can be modeled with multiple 2- or 3-dimensional functions. The ambiguity shape can represent a surface, a closed or solid shape or an open or inverse solid. This enables multiple ambiguity shapes to intersect to determine phenomenon spatial constraints more accurately.

6.22.7.5 Array Values

Parameter of functions or characteristics (such as integration time) may be dependent on their position within the TCF array. An example of this is when a CCD array must be radiometrically balanced to obtain uniform intensity response across the array. Position and times of data within a dataSet within a TCF have coordinates unique to their position within the TCF. The array values will have the same number of values as cells or indexes in the TCF array. The array values will have a one-to-one correspondence to cells in the TCF, which lead to the data.

6.23 Process components

Transducer data may be processed or may need to be processed for a variety of reasons. When data is received from a transducer the data may have been processed by one or more processes. Alternatively, when data is sent to a transducer it may go through one or more processes prior to the transducer receiving it. It would be ideal if a process could be modeled, such that if the process model and input are known, then the output could be determined. This would be necessary for determining what the data would look like in the transmitter side. Alternatively, it would be ideal if a process could be modeled such that if the process model and output data are known then the input could be determined. Examples of transducer processes are integration, differentiation, averaging, compression, encoding, frequency mixers, filters, discriminators, etc. A system may be composed of a number of transducers and processes. Each transducer and process is described as a stand-alone entity or component. These components can be reused in other systems as deemed necessary.

6.23.1 Process Modeling

Process modeling is very difficult. There are various types of processes that come into play with transducer data and TML. Processing of vectors, signals, images, and data are all possibilities of types of processing that may be required. Various properties of processes are linearity, time invariance, memory, reversibility, etc. For the current time a process can only have a single output. Multi-ended processes must be described with separate processes. Currently, the way TML models processes is to simply give it a unique URI identifier, a name, identify what the inputs and output are, and describe any parameters which are unique to that particular process. The URI may point to a site on the network where the process algorithm is defined in more detail.

6.24 System component relationships

The systems relations data takes a higher-level view of the system. The transducers and processes in the system are treated as components. The relations data describes relationships among the various components in the system as well as to external datum's. One of the keys to sensor fusion is to have data represented in a common datum. TML tries to impose the linkages of data to standard, space, time and measurement datums.

The transducer and process descriptions are descriptions that an original equipment manufacturer can develop at the time of manufacture. The system relationships description is usually developed after the transducer and process components have been integrated into a system. When the TML document is developed for the system the process and transducer components can be plugged into the system and the relations relationships can be added without changing the previously developed component descriptions.

6.24.1 Position

The position relation identifies the location and attitude of a transducer relative to a spatial reference frame. The transducer may be positioned relative to another transducer or it may be positioned relative to a spatial datum such as ECEF. The position of a transducer may be a time dependent position. If so, then the position will be the result of a sensor measurement.

6.24.2 Absolute time

The normal way for TML to capture data is to capture it in TCFs and tag those TCFs with a time tag, which is typically the state of the system clock at the instant of the transducer TCF trigger. This relative time tag will enable all of the TCFs from all of the transducers in a system to be aligned in time relative to one another. However, at this point the real or absolute time of any of the transducer events is still unknown. The time datum dependency identifies where the absolute time references comes from. This relation will identify which sensor dataUnit measures the date and time.

6.24.3 Connect

The connectivity or functional flow of data is described so that by the time data is received it may have gone through several processes. In order to understand what to do with the data we must know its processing heritage. For example, if data has been compressed then Base64 encoded, the order and the process the original data went through must be known in order to get to the needed data. The connect relationship enables transducers and process inputs and outputs to be connected together to describe the data flow through the system.

6.24.4 Remote energy

Remote transducers rely on energy transfer to detect or influence objects at a distance. The medium of propagation is an important consideration to understand precisely the effects on the energy as it is sensed. For example, energy propagates differently through water than air, and proper understanding of the sensor data relies on a knowledge of the medium through which the energy was transmitted.

6.24.5 Feature of interest

The feature of interest would probably be better suited within the transducer model except for that many times the feature of interest is not known until the transducer is installed or after interpretation of the data. For this reason it was added to the relationships to signify to external nature to the transducer component. Often it is just a relationship of just one feature to one data. However the relationships may become a little more complex. TML handles the complexity of many-to-one or a one-to-many relationship between transducer data and features.

Features or objects are things (matter or energy) in the real world. Features exhibit their qualities and quantities through properties of phenomena. Sensors are able to detect the phenomena properties and processors and data exploiters are able to make predictions of what the object is by interpreting the phenomenon. The more phenomena from an object the better the prediction of what the object is.

In some cases the object is known at the time of system integration. The phenomenon can then be associated with an object at that time. For example, a pressure sensor may be installed in a pressure boiler. The phenomenon property of pressure would relate to the feature or object “pressure boiler”.

If features or objects are being investigated purely as a result of interpreting the remotely sensed data, then it is certainly required to have an understanding of the propagation mechanisms at work. This knowledge is acquired from outside of the TML domain of supplied data. For example, a radar system looking out several thousand kilometers relies on the refraction of the atmosphere to bend the RF energy around the earth’s surface to enable them to see close to the surface at long distance. The TML characterization would only describe the radar as if it were operating in a vacuum. The TML descriptions are and must be spatially and temporally invariant. A system designer could use sensors to characterize the propagation of energy through a medium, if they did not want to leave it up to the data interpreter.

Example 1:

The following example shows how TML would describe a wind anemometer with two data units in a single reading, including wind speed and wind direction.

```
<logicalDataModel>
   <dataSet id="WIND_READING">
      <dataUnit id="WIND_SPEED">
         <name>Wind Speed</name>
```



```
            <data_type>number</data_type>
         </dataUnit>
         <dataUnit id="WIND_DIRECTION">
            <name>Wind Direction</name>
            <data_type>number</data_type>
         </dataUnit>
      </dataSet>
</logicalDataModel>
```

Example 2:

The example below is the model for a simple camera with separate Red, Green and Blue pixel values. The camera image is 640 x 480 pixels.

```
<logicalDataModel>
   <dataArray id="CAMERA_ROWS">
      <arraySize>480</arraySize>
      <dataArray id="CAMERA_COLS">
         <arraySize>640</arraySize>
         <dataSet id="CAMERA_PIXEL">
            <dataUnit id="CAMERA_PIXEL_RED">
               <data_type>number</data_type>
            </dataUnit>
            <dataUnit id="CAMERA_PIXEL_GREEN">
               <data_type>number</data_type>
            </dataUnit>
            <dataUnit id="CAMERA_PIXEL_BLUE">
               <data_type>number</data_type>
            </dataUnit>
         </dataSet>
      </dataArray>
   </dataArray>
</logicalDataModel>
```


7 TML Structure (Normative)

TML provides a self-contained lightweight XML envelope for efficient transport of transducer data as well as all necessary information to decode, process, analyze and understand that data. Transducer system, spatial and temporal metadata are also characterized and carried along with the data, allowing analysis of transducer data and processing of the data only using information carried in the TML data stream.

TML achieves interoperability through several key features. These include:

- ability to plug TML behavior and response models into a standardized transducer definition
- ability to interface with and utilize arbitrary sensor output
- normalization of discordant datasets using TML behavior models
- utilization of GML and possibly SensorML base types (future).

TML is composed of overall identifications, transducer descriptions, clock descriptions, process descriptions, and relations between components of transducer systems. Figure 34 below shows the top level elements of TML.

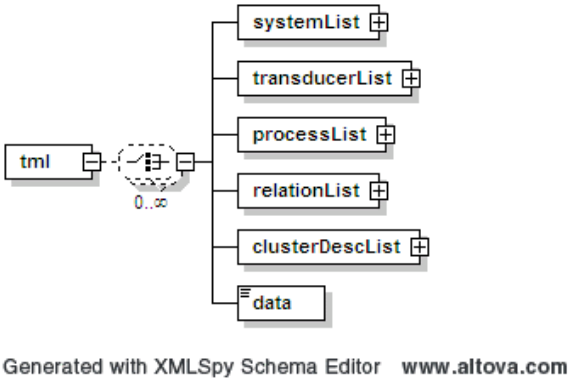


Figure 34 - TML Top Level Elements

Figure 35 below illustrates the overall structure of TML, including the system, transducer, process, relation, data cluster and data models. Note that this is not necessarily a complete TML model, and is intended only for illustrative purposes to show the overall structure of a TML description.


```
<?xml version="1.0" encoding="UTF-8"?>
<tml          version="1.0.1          20060301"          xmlns="http://www.opengis.org/tml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.org/tml ..\tml-new-framework-20060301.xsd">
    <systemList>
        <system id="SYS01">
            <sysClk id="SYS01_SYSCLK">
                <period>
                    <value>0.001</value>
                </period>
            </sysClk>
            <transducerList>
                <transducer id="SYS01_CAMERA">
                    <logicalDataModel>
                        <dataSet id="CAMERA_IMAGE">
                        </dataSet>
                    </logicalDataModel>
                </transducer>
            </transducerList>
            <processList>
                <process id=”PROCESS_1” ref=”SYS01_PCS”>
                    <input>
                        <dataSet id="CAMERA_IMAGE">
                        </dataSet>
                    </input>
                </process>
            </processList>
            <relationList>
                <timeDatum sysClkRef="SYS01_SYSCLK" ref="SYS01_GPS" timeDatum="UTC">
                    <year bindUIDref="GPS_TIMESTAMP_YEAR"/>
                    <month bindUIDref="GPS_TIMESTAMP_MONTH"/>
                    <day bindUIDref="GPS_TIMESTAMP_DAY"/>
                </timeDatum>
                <position transducerUIDref="SYS01_CAMERA" relToCompUIDref="SYS01_GPS"/>
                    <posCoord coordName="lat" uom="radians">
                        <value bindUID="SYS01_GPS_LAT"/>
                    </posCoord>
                </position>
                <attach bindUIDref="SYS01_GPS_LAT">
                    <value bindUIDref="GPS_LATITUDE"/>
                </attach>
            </relationList>
        </system>
    </systemList>
    <clusterDescList>
        <clusterDesc id="SYS01_GPS_LLA">
            <encoding>
            </encoding>
        </clusterDesc>
    </clusterDescList>

    <data clk="263084829" ref="SYS01_TIMESTAMP">2006-03-02T14:39:41.04Z</data>
    <data clk="263085859" ref="SYS01_GPS_LLA">0.577041 -2.035342 0.000000</data>
    <data        clk="12344"        ref="SYS01_CAMERA_RGB24_BASE64">ABabdks2836875........
ARBKDK==</data>
</tml>
```

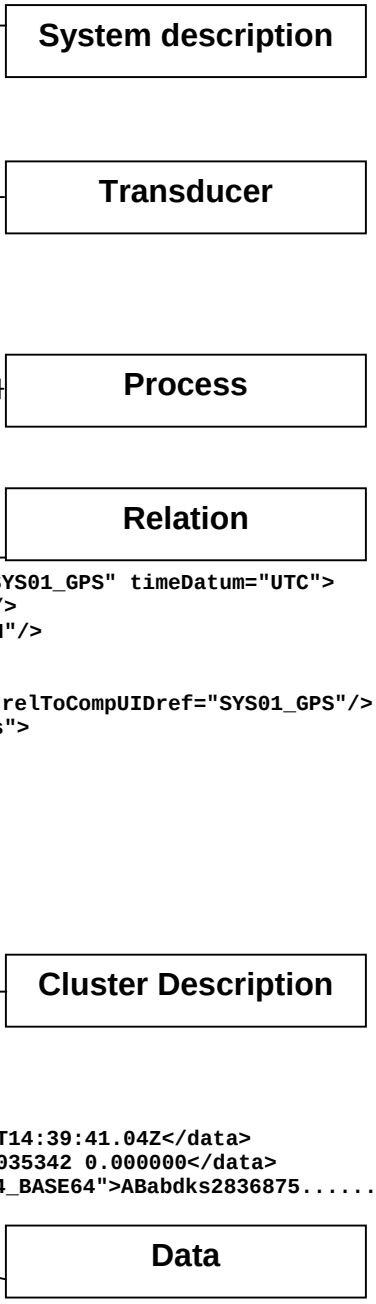


Figure 35 - TML Models

The TML transducer logical data model and response models could harmonize with other standards such as GML, Observations and Measurements and SensorML by utilizing relevant base constructs. Figure 36 below displays how TML could work with other standards.

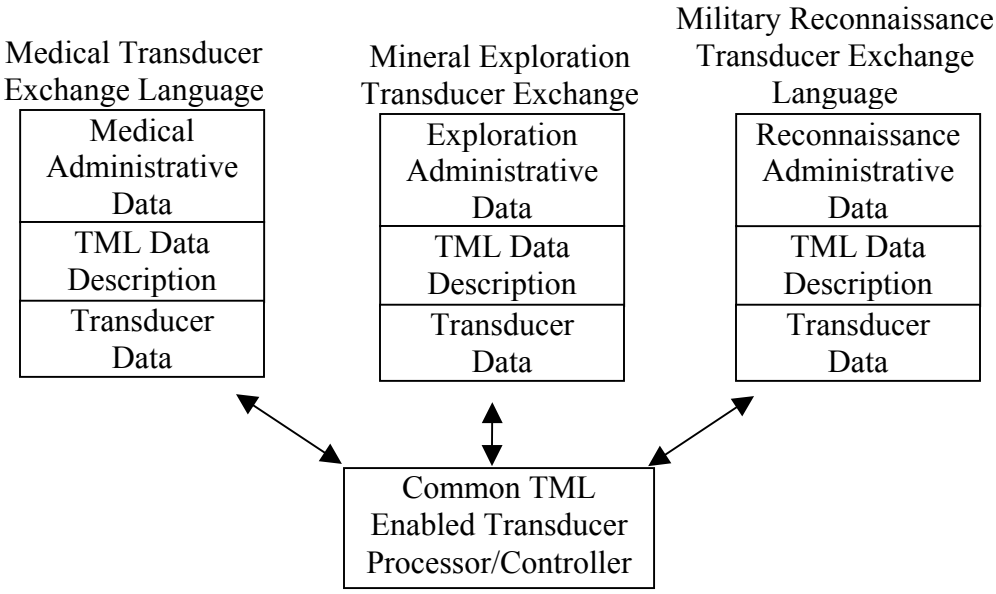


Figure 36 - Sharing of transducer data across application domains

Other than a general philosophy of 'plug and play' and 'composable' systems, TML is designed to be independent of object, pattern or process design methodologies.

TML transports arbitrary data formats from a sensor platform because it is not always practical to adopt a standardized data encoding. The following examples illustrate some cases where this is of use:

- Many sensor manufacturers already have arrangements with third party software firms to design the output format.
- Changing the output format of a sensor may impact existing deployments.
- Sensors often come with software packages meant to act as the client itself, or as the interpreter of the data format.
- Sensor data may have security restrictions requiring a private key mechanism for decoding. In this case the format cannot be revealed for purposes of developing application schemas.

A high resolution timestamp is attached to every TML data cluster. This permits precise determination of when a transducer measurement was taken, and allows the temporal alignment of data from many sources. This also permits the precise reconstruction of events as recorded by systems of transducers.

It is paramount to preserve raw transducer data in as close a manner to the original form as possible. Sensor data is often an artifact of the sensor's internal processing rather than a true record of phenomena state. For example, radar systems frequently capture a magnitude image but all of the phase information from the received waveform is absent from the magnitude image. Some applications require only the radar magnitude image

while others require both magnitude and phase data. Situations such as these make it prudent to retain original unaltered data whenever possible prior to any processing.

Other situations may require preservation of original data as well as modification of the original output or the interpolation between successive transducer measurements to determine a value at a particular time. TML's response models allow for the capture of original data and the characterization of data as close to the original form as possible.

TML is designed for delivery of data from live sensor platforms. In the sensor-to-client communications path, TML generally precedes translation to richer semantic encodings such as GML or Observations and Measurements. The latter encodings are more geared toward data exchange or scenarios in which Filter Encoding queries may be desired.

As new taxonomies are developed in the geospatial arena the need for domain agnostic representations like TML becomes greater.

7.1 TML Data Models

> TML defines data as follows: Data is the response resulting from a phenomenon acting as a stimulus for a receiving or sensing type of transducer. The data will be some function g(f(x,y,z,t)) of the phenomenon. In addition, data can be the stimulus to a transmitting or actuating type of transducer where the response will be a phenomenon. The data may represent a function of location and time [g(x,y,z,t)].

TML data is carried in data clusters, marked with a "data" tag. This includes attributes for a local clock value and reference identifier for the transducer descriptions (Behavioral and Data Models).

```
<element name="data" type="DataClusterType"/>
<complexType name="DataClusterType">
   <simpleContent>
      <extension base="string">
         <attribute name="ref" type="Name"/>
         <attribute name="clk" type="integer"/>
         <attribute name="dateTime" type="date"/>
         <attribute name="reference" type="anyURI"/>
      </extension>
   </simpleContent>
</complexType>
```

The tml:data element includes the attributes of ref, clk, dateTime and reference. Ref is a short local reference to a Data Cluster Description (the Behavioral and Data Models). Clk is an integer number which is the value of a local stable counter associated with the enclosed data. DateTime is an optional attribute that can be used to associate a fully qualified date timestamp with a data cluster. The recommended format is "CCYY-MM-

DDThh:mm:ss.sssZ". Reference is a fully qualified URI to the Data Cluster Description, System and Transducer.

The time tag is particularly important to TML, as it allows precise determination of when a transducer measurement was taken. This time reference is put into the data cluster and represents the time at which the trigger occurred to either sample data (for receivers) or act on data (for transmitters). In the case of transmitters, the time tag may represent a time in the future. This is meant for the transducer system to act at a future time. If no time tag is present in data for a transmitter, then the action shall occur as soon as possible.

Below is a sample data tag as well as examples.

```
<data dateTime="2006-02-01T14:41:31.01Z"
reference="FULLYQUALIFIED_URI_SYS01_TEMP">223</data>
```

Example 1:

This example shows a data cluster at system clock time '12344' from a transducer with transducer id 'SYS01_TEMPERATURE_ASCII'.

```
<data clk="12344" ref="SYS01_TEMPERATURE_ASCII">223</data>
```

Example 2:

This example shows a data cluster at system clock time '88734' from a transducer with transducer id 'SYS01_CAMERA_RGB24_BASE64'.

```
<data clk="12344" ref="SYS01_CAMERA_RGB24_BASE64">ABadk26175....ARBKDK==</data>
```

Example 3:

```
<data clk='28118792' ref='SYS01_TEMPERATURE'>21.1</data>
<data clk='28118795' ref='SYS01_TIMESTAMP'>2005-08-26T16:31:49Z</data>
<data clk='28118800' ref='CAM'>AB12...base64 RGB24 image ...12Aa=</data>
<data clk='28118874' ref='SYS01_WIND'>9.8 3.1415</data>
```

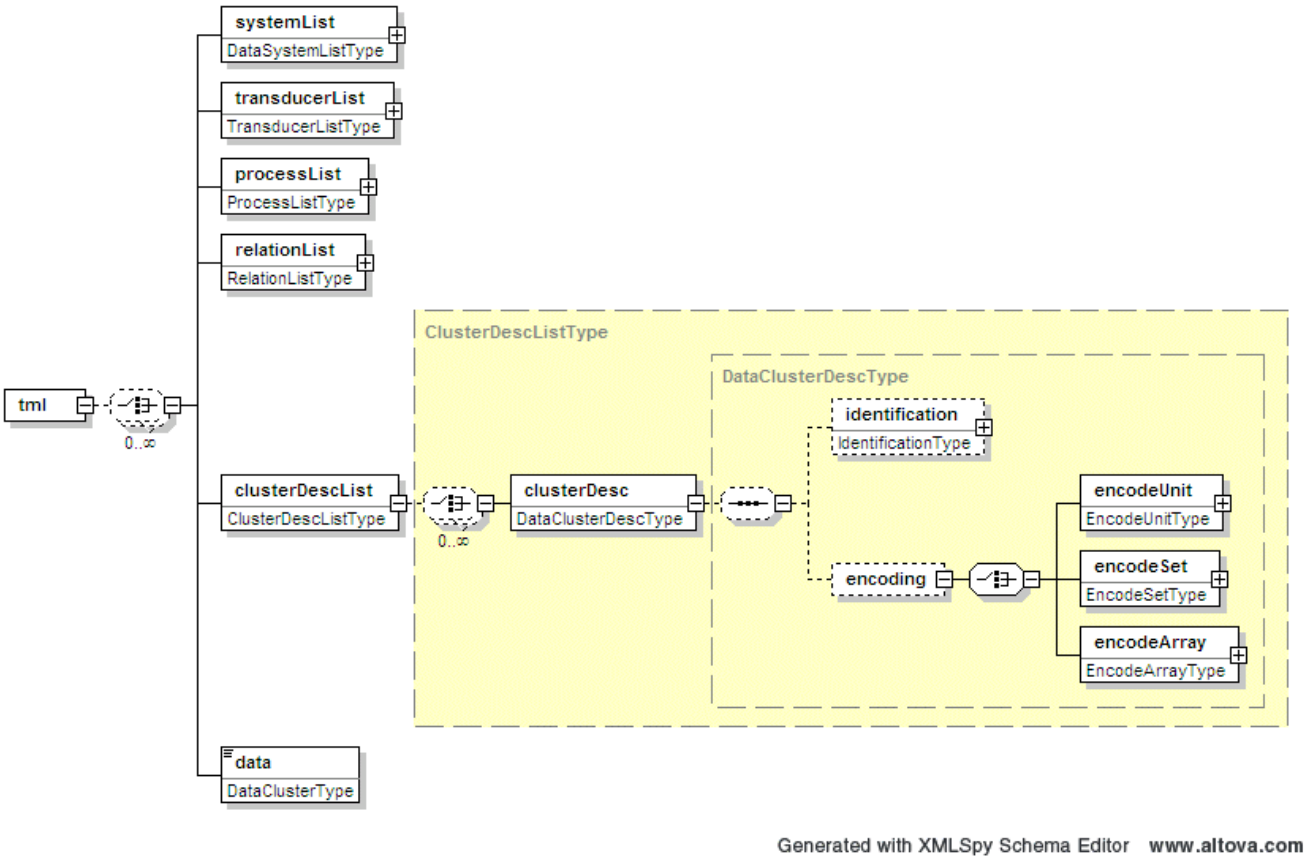



Figure 37. clusterDescList Top Level Diagram

7.2 TML Data Clusters

The element tml:clusterDescList includes all elements below.

7.2.1 tml:clusterDesc

The TML cluster description describes the implicit structure of raw data within a TML cluster. It represents a mapping from the logical data model to raw data and provides the encoding/decoding information to enable the understanding of the transducer data. The clusterDesc tag includes the attributes id and ref.

Data blocks can be described in four ways within TML. They include:

- Text character delimited
- Text width delimited
- Binary Base 16 encoded
- Binary Base 64 encoded

The Data Cluster Description describes and provides the implicit structure of the data message that is transmitted in a data cluster. The Data Cluster Description has an

identifier and the encoding element which correlates to a data model and specifies the implicit structure of the data message. The specifics of how the implicit structure is encoded is in two parts, the first part is the use of the TCF, sample and measurement layout, and the second part is the encoding details (specified in encodingDetails). The encodingDetails specify the information necessary to split the data out into the corresponding data fields as specified in the Data Model.

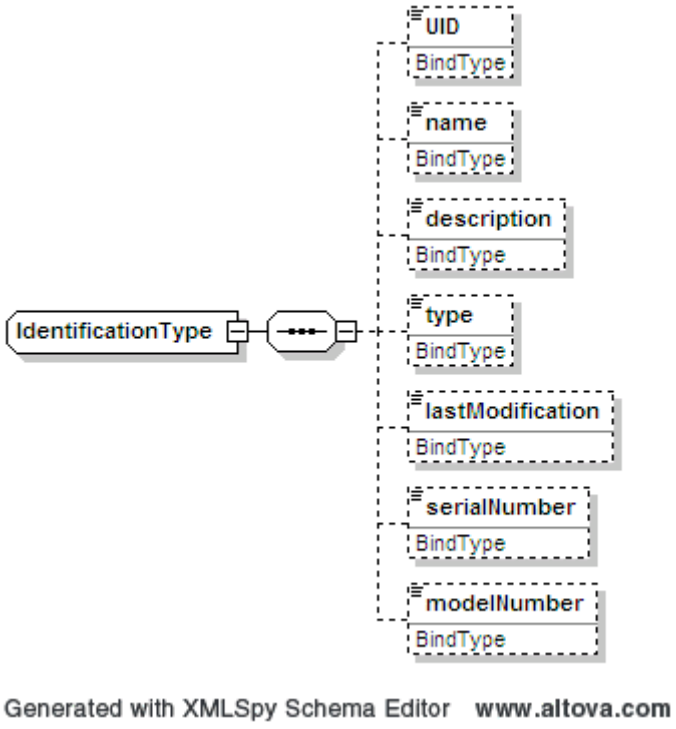


Figure 38. Identification Type Diagram

7.2.1.1 tml:identification

This is an identifier unique to this particular element. The identification tag includes the elements UID, name, description, type, lastModification, serialNumber and modelNumber. All of these elements include the attribute bindUID. UID is the Universal Identification Number of the system, transducer, process, relation or data cluster. The identification element type is repeated throughout the schema to uniquely identify anything that may need identifying.

7.2.1.2 tml:encoding

This specifies which encoding variant is to be used for this raw data block (“binary_base64”, "binary_base16", "text_delimited", "text_fixed_width").

The attribute of encodingType in the encoding element specifies which basic format the data block is in. It also specifies the format and required information of the encodingDetails elements. The attribute of complexity is described below.

Below is the table of attribute values for encodingType to the xml field type for the associated data cluster.

Table 4 – encodingType attribute list and associated xml primitive data type

Attribute value	XML Field type for associated data cluster
binary_base64	xs:base64Binary
binary_base16	xs:hexBinary
text_delimited	xs:string or xs:normalizedString
text_fixed_width	xs:string or xs:normalizedString

Another TML concept is the complexity level, a numerical value representing the complexity of a TML message. The numerical value will indicate what processing resources and communication resources are required to adequately handle the TML message.

Table 5 - Complexity levels

Memory Requirements		**<100 MB**	**100 MB - 1 GB**	**1 GB - 10 GB**	**10 GB - 100 GB**	**>100GB**
Real-time FLOPS		**<10 MFLOPS**	**10 MFLOPS - 100MFLOPS**	**100 MFLOPS - 10 GFLOPS**	**10 GFLOPS - 1 TFLOPS**	**> 1 TFLOPS**
File Size	**Bandwidth**					
<100KB	**<100Kb/s**	1A	1B	1C	1D	1E
100KB - 1MB	**100Kb/s - 1Mb/s**	2A	2B	2C	2D	2E
1MB - 10MB	**1Mb/s - 10Mb/s**	3A	3B	3C	3D	3E
10MB - 1GB	**10 - 100Mb/s**	4A	4B	4C	4D	4E
>1GB	**>100Mb/s**	5A	5B	5C	5D	5E

FLOPS: FLoating point Operations Per Second

7.2.1.2.1 tml:encodeUnit

The element encodeUnit has the attributes of id, ref and encodingType.

The encodeUnit is the smallest encoding block for describing the implicit structure of data.

7.2.1.2.1.1 tml:name

The name is a human readable identifier.

7.2.1.2.1.2 tml:indexSequence

This has a bindUID attribute.

7.2.1.2.1.3 tml:encodingDetails

The following tables illustrate the various possibilies for use in encoding both ASCII and binary data within TML.

Table 6 – Ascii Encoding data types

Encoding data type	Explanation
Raw	Raw data block
Number	General Number in recognized fomats
Integer number	Integer numbers only no floating point
Real number	Floating point number
String	Simple text string

Table 7 – Binary Encoding data types

Encoding data type	Explanation
integer signed integer	fieldWidth specifies the size of the integer 2's complement signed integer
unsigned integer	Unsigned integer where fieldWidth specifies the size
IEEE Float	Standard IEEE 754 single precision 32 bit

IEEE double	Standard IEEE 754 double precision 64 bit
raw	Raw data block
string	Simple text string

7.2.1.2.1.3.1 tml:fieldOffset

The fieldOffset element includes the attribute offsetRef, and includes the elements chars, bytes and bits. Chars, bits and bytes are bindUID types. The fieldOffset is the offset of the field in the implicit structure.

7.2.1.2.1.3.2 tml:fieldWidth

Includes the elements chars, bytes and bits. The fieldWidth is the width of the field in the implicit structure.

7.2.1.2.1.3.3 tml:textFieldType

Includes the attribute of beginningDelimiter. The textFieldType should be picked from Table 6. BeginningDelimiter identifies the delimiter for the start of the data field.

7.2.1.2.1.3.4 tml:binaryFieldType

Includes the attribute of endian. The binaryFieldType should be picked from Table 7. Endian can be either big or little.

7.2.1.2.2 tml:encodeSet

The encodeSet tag includes the attributes of id, ref and encodingType, as well as the elements name, indexSequence, encodeUnit, encodeSet and encodeArray. The encodeSet specifies a portion of the implicit structure of the data.

7.2.1.2.3 tml:encodeArray

The encodeArray tag includes the attributes of id, ref and encodingType, as well as the elements name, arraySize, indexSequence, encodeUnit, encodeSet and encodeArray. The encodeArray specifies a portion of the implicit structure of the data.

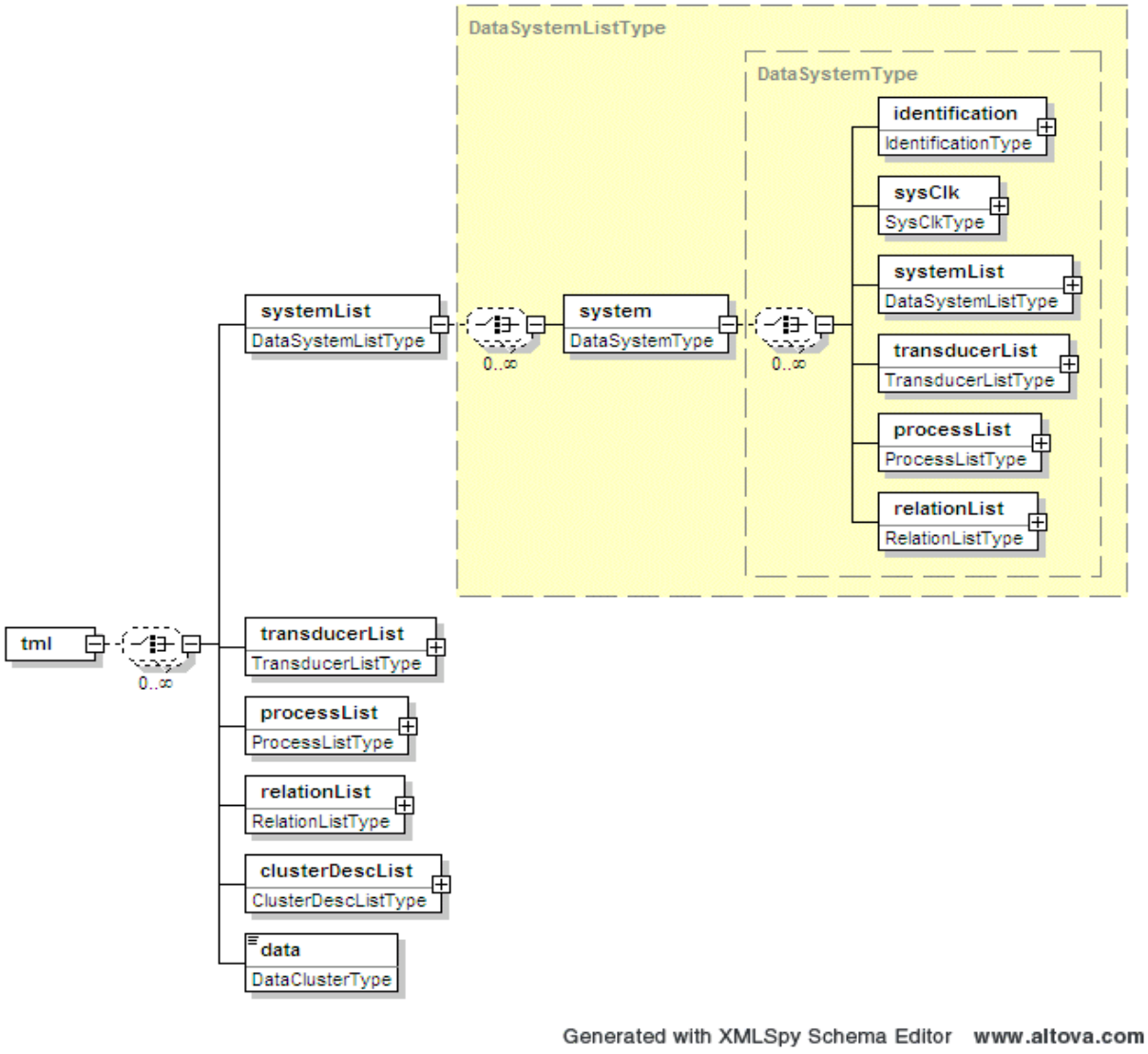


Figure 39. systemList Top Level Diagram

7.3 TML System

The tml:systemList tag includes all the elements described below.

7.3.1 tml:system

The system tag includes the attributes of id and ref. A system is a functional system composed of one or more transducers and/or processes. The data system view of a transducer system describes how to understand the transducer data. It describes the transducer organization of data, the transducer response and geometry characteristics, the

pre-processing between the data and the transducer, and finally it describes the external logical relationships relating to understanding how multiple components (transducers and processes) within a system relate to one another.

Transducer systems are a collection of transducers and processes working together to produce TML data or to utilize TML data to invoke desired phenomenon states. A transducer system is required to produce or act on TML data, such that a transducer system is either a source or a sink for transducer data. By definition a transducer or process does not produce or accept TML data. If it does, then by definition it is a system of one transducer or process. A transducer system will contain one or more transducers and/or processes and a *TML transducer* or *process node*. Transducers in a system may be independent or complementary (provide supporting measurements for other transducers).

EXAMPLE: a complementary set of transducers: a GPS provides position and an IMU provides attitude for a camera.

7.3.1.1 tml:identification

See section 7.2.1.1.

7.3.1.2 tml:sysClk

Includes the attributes id and ref, as well as an identification element and the elements described below. The sysClk is described in Section 6.

7.3.1.2.1 tml:period

Includes value and accuracy elements. Value is a bindUID and accuracy has the attributes of bindUID, type and factor. The TML user can choose to define the system clock using period or frequency.

7.3.1.2.2 tml:frequency

Includes value and accuracy elements.

7.3.1.2.3 tml:minCount

The minCount element has a bindUID attribute. The minCount specifies the minimum value that a system clock has.

7.3.1.2.4 tml:maxCount

The maxCound element has a bindUID attribute. The maxCount specifies the maxium value that a system clock has.

7.3.1.3 tml:systemList

See section 7.3.

7.3.1.4 tml:transducerList

See section 7.4.

7.3.1.5 tml:processList

See section 7.5.

7.3.1.6 tml:relationList

See section 7.6.

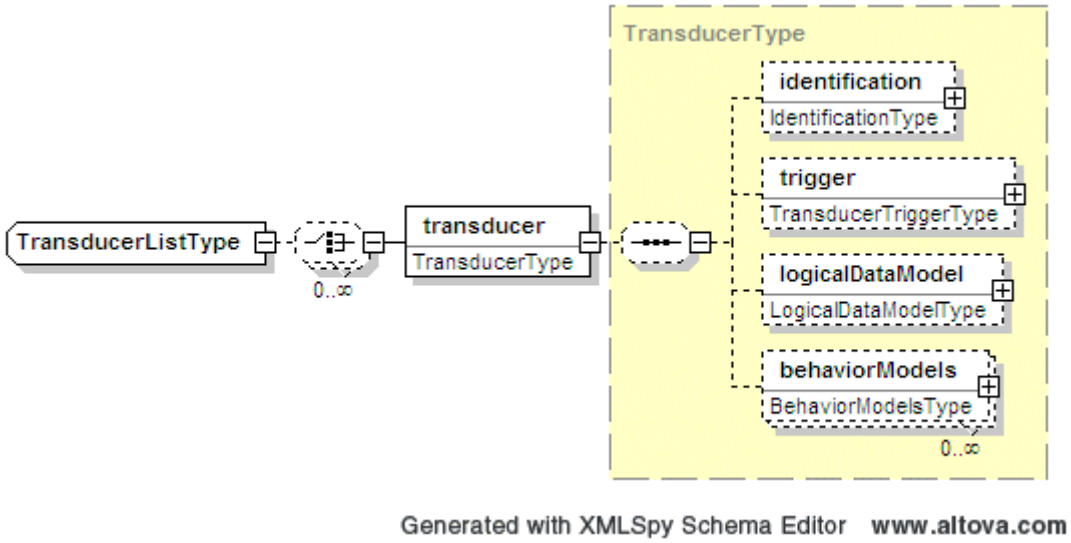


Figure 40. Transducer Top Level Diagram

7.4 TML Transducer

The tml:transducerList tag includes the elements described below.

7.4.1 tml:transducer

Includes the attributes of id and ref, as well as an identification element and the elements below.

For this specification, the definition of a transducer is limited to be a physical or virtual entity capable of translating between physical phenomenon and transducer data. Transducers that translate phenomenon to data are typically referred to as receivers or sensors, and transducers that translate data to phenomenon are typically referred to as transmitters or actuators. Transducers are the interface between the digital realm and the physical realm or real world. Any time a computing machine is processing data, its input is received through a transducer and its output is through a transducer.

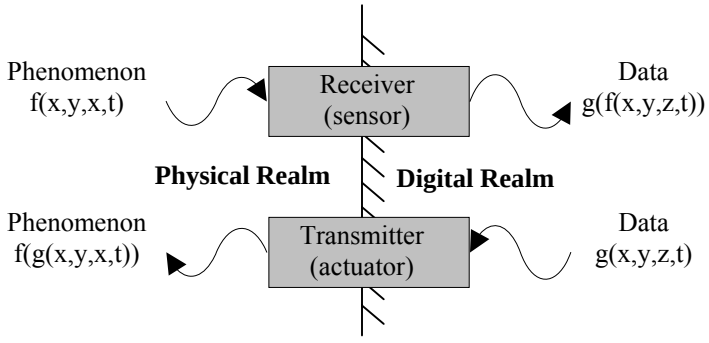


Figure 41 - transducer reference model

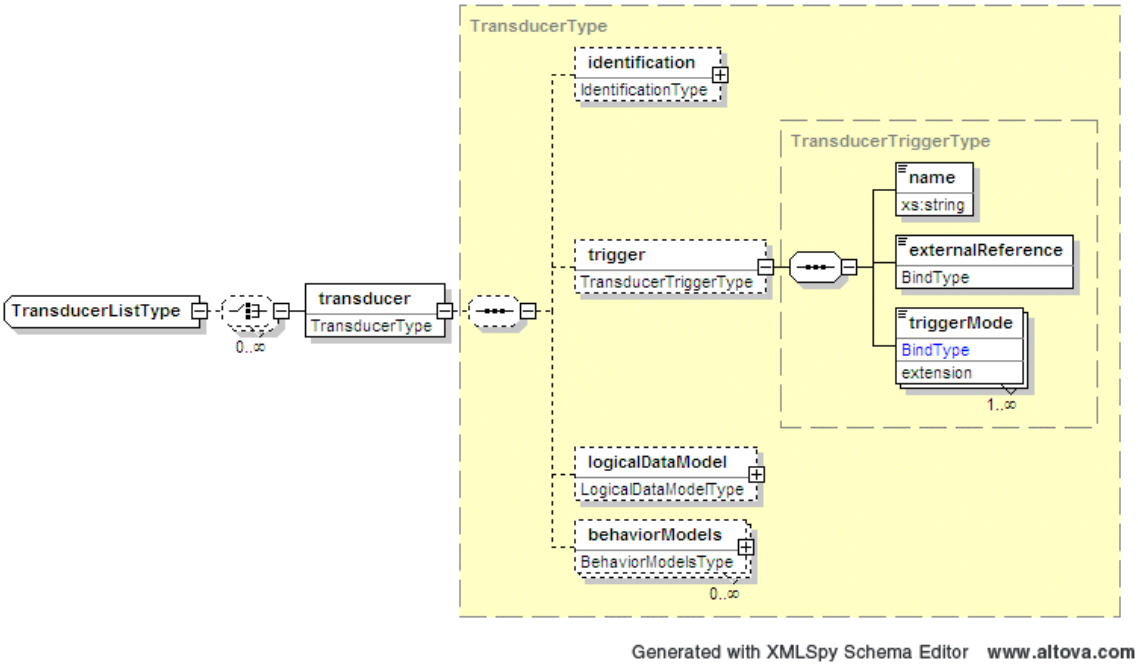


Figure 42. Transducer trigger diagram

7.4.1.1 tml:trigger

A signal that initiates a transducer event. A trigger may be generated internally (either periodically or randomly) or an external trigger can initiate a transducer event.

7.4.1.1.1 tml:name

Name is a human readable identifier.

7.4.1.1.2 tml:externalReference

The externalRefernce element has a bindUID attribute. The externalReference identifies the external reference signal that is used as a trigger for this transducer.

7.4.1.1.3 tml:triggerMode

Includes the attributes of bindUID and triggerType. The triggerType specifies the trigger type, whether it be rising edge, falling edge, rollover, match or modulus. The triggerMode defines the type and the UID.

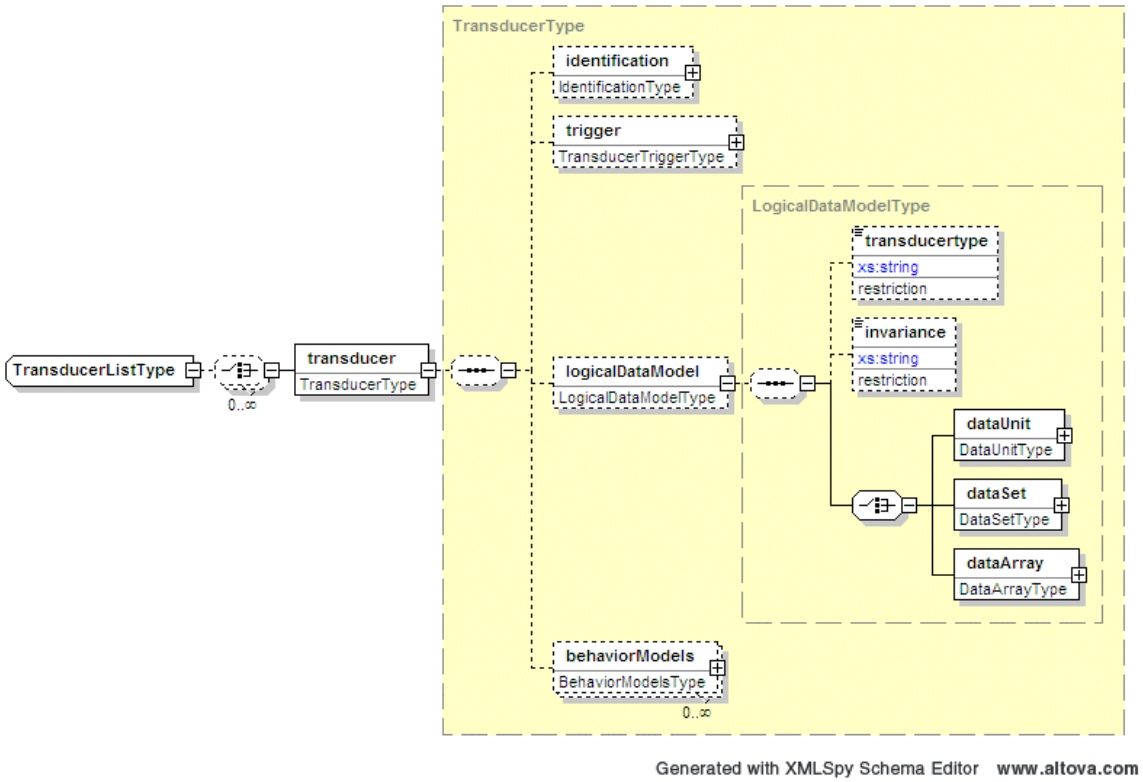


Figure 43. Transducer logicalDataModel diagram

7.4.1.2 tml:logicalDataModel

The logicalDataModel defines the layout of data for later processing it. For example, data can be laid out in line scans, or an array, or other ways.

7.4.1.2.1 tml:transducerType

The transducerType can be either insitu or remote.

7.4.1.2.2 tml:invariance

Invariance can refer to both spatial and temporal invariance. Spatial invariance refers to the fact that the spatial characteristics of a model are invariant anywhere in the world, while temporal invariance mean that the temporal characteristics of a model are the same regardless of when they are observed.

7.4.1.2.3 tml:dataUnit

Includes the attributes of id and ref, as well as the elements name, indexSequence and data_type. The tag indexSequence is a bindUID.

A dataUnit contains the description for the individual data value corresponding to an instance of a phenomenon. One or more data units are contained in a data set. As

transducer data gets further from the transducer (i.e. processed), it may be necessary to handle the data unit as a digital blob where it no longer relates directly to a phenomenon. To understand how it indirectly relates it would be necessary to follow the process path to the phenomenon.

7.4.1.2.4 tml:dataSet

Includes the attributes of id and ref, as well as the elements of name, indexSequence, dataUnit, dataSet and dataArray.

A dataSet is a set of heterogeneous data units, or data groups. One or more data sets will compose a data array. In addition, one or more data units will compose a data set.

7.4.1.2.5 tml:dataArray

Includes the attributes of id and ref, as well as the elements of name, arraySize, indexSequence, dataUnit, dataSet and dataArray.

A dataArray is a homogeneous set of *data sets*. In some cases, a data array can contain a set of homogeneous data arrays. A data array is composed of one or more data sets or data arrays. A specific data array is the TCF data array.

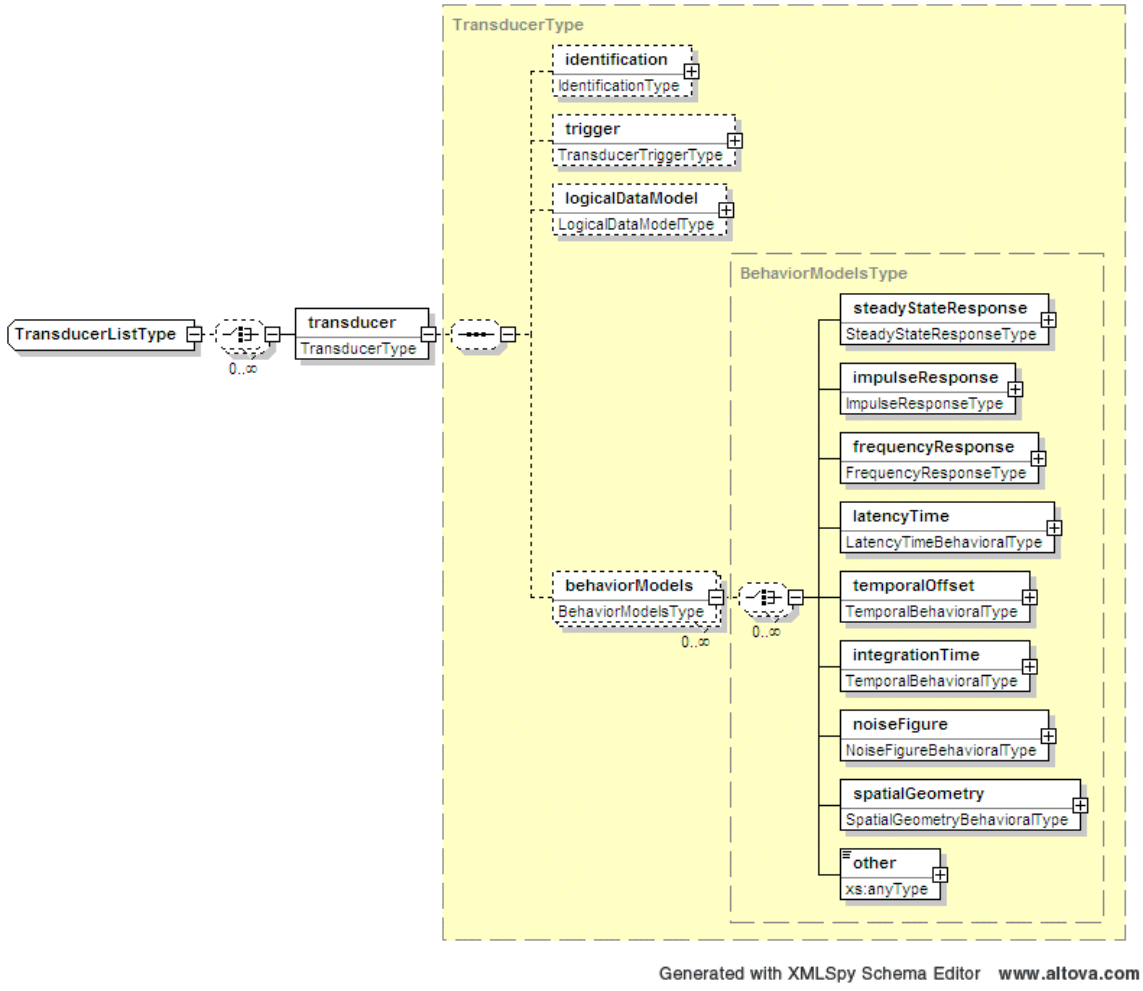


Figure 44. Transducer behaviorModels diagram

7.4.1.3 tml:behaviorModels

The behaviorModels element includes the attribute of dataUnitUIDRef.

Sensor data is an artifact of the sensor's internal processing of sensed phenomena. The effects of this processing on sensed phenomena are hardware-based and can be characterized. This processing reflects hardware limitations of that sensor and these limitations may cause the data to be different.

TML behavior models are formalized XML descriptions of known sensor hardware processes. They can be applied as correction factors to bring sensed artifacts back into the phenomena realm. As these artifacts move closer to the phenomena realm and achieve a higher level of accuracy they bring richer information to fusion applications.

In general response models are useful to any application that requires a very high level of data fidelity.

Normalization information for a transducer is captured and documented in TML. The data itself is not required to be normalized. To illustrate the value of normalization, assume several separate and independent groups are all collecting weather information.

The first group is a weather forecasting team which logs their information into a database. Their desktop client interfaces with a web service which provided temperatures in Degrees F. They log these Fahrenheit values and their client displays Fahrenheit values.

The second is a school study group obtaining and logging temperatures in Degrees C. Similarly to the first group, this group's client displays Celsius values.

A third group logs thermocouple readings for their temperature data.

If we want to create a single client to interface with these three data sources it would have to know all associated units of measure as well as how to convert them to a common unit of measure. TML simplifies the problem by provides normalization information along with the raw data that is captured.

All behavior models are applied to the atomic measurement, that is, the individual number representing a particular transducer output value. Each response parameter is associated with this transducer observation in one of two ways:

- *static* response parameter: an unchanging single value or unchanging single array of values associated with each atomic observation in the primary data stream. This can be sent once at the start of a data transmission and applied to the primary data at the receiving end.
- *dynamic* response parameter: a changing single value or changing single array of values associated with each atomic observation in the primary data stream. Dynamic association requires a separate stream to carry the changing response parameter. This stream is described in TCF format as coming from a virtual sensor.

Behavioral models also include characterizations of transfer functions. These enables an output to be determined if the input and transfer functions are known or the input can be reconstructed from the output and the transfer function. The transfer function characterizes both the static (steady state) and dynamic (impulse response) characteristics of a transducer and/or process. The impulse response function is only applicable to *linear time invariant* transducers and/or processes.

7.4.1.3.1 tml:steadyStateResponse

The steady state transfer function provides a mapping of the phenomenon to the data for the steady state. This function does not characterize dynamic characteristics of the transducer. This function does however characterize linear and non-linear regions of the transducer.

A steady state transfer function maps data to phenomena property. The transfer function is much like a dynamic lookup table in TML. For calibrated response transducers the steady state transfer function will describe the mapping between the data and the phenomenon. The accuracies as well as the allowed data values can be expressed in the transfer function as well. Many sensors are not calibrated. They make relative responses based on a previous history and there response can not be traced to calibrated measurement standards. For these un-calibrated transducers the full steady state transfer function does not exist. Description may exist such as just the phenomenon or the data description.

A steady state response brings data to a state independent of the units of measurement.

Figure 45 is an example of a steady state transfer function. In general, the transfer function shows the dynamic range, sensitivity and saturation levels for the data, any dead zones, linearity of the response, and quantization levels. The transfer function description is also useful in defining allowed data values; the transfer function may also be dynamic or changing as various conditions change. The transfer functions are described normalized with gain and bias factors for each axis. The gain and bias parameters that that are used to describe the transfer function can also be derived from sensor readings themselves.

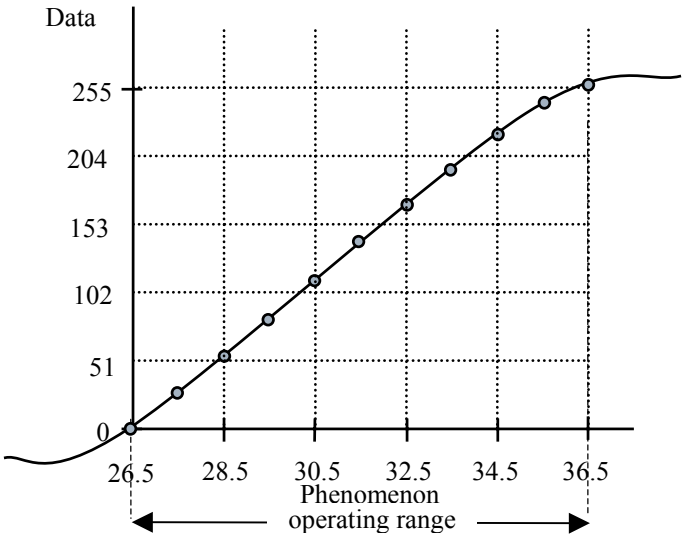


Figure 45 - Example steady state transfer function

Example 1:

This example shows a temperature sensor with data reported back in degrees Kelvin.

```
<steadyStateResponse>
   <phenAxis units="kelvin" direct="true">
      <phenType>Temperature</phenType>
   </phenAxis>
   <dataAxis direct="true"></dataAxis>
</steadyStateResponse>
```


Example 2:

This example shows a temperature sensor with data reported back in degrees Fahrenheit. Note the use of the multiplier and offset.

```
<steadyStateResponse>
   <phenAxis units="fahrenheit" hysteresisDirection="both"  direct="true">
      <multiplier>1</multiplier>
      <offset>273.15</offset>
      <phenType>Temperature</phenType>
   </phenAxis>
   <dataAxis hysteresisDirection="both" direct="true">
      <multiplier>0.555556</multiplier>
      <offset>-32</offset>
   </dataAxis>
</steadyStateResponse>
```

Example 3:

This example shows the steady state response of a temperature sensor with data reported back in degrees Kelvin as an unsigned integer with the following transfer function:

```
<steadyStateResponse>
   <phenAxis units="kelvin" hysteresisDirection="both"  direct="true">
      <multiplier>1</multiplier>
      <offset>273.15</offset>
      <phenType>Temperature</phenType>
   </phenAxis>
   <dataAxis hysteresisDirection="both" direct="true">
      <multiplier>0.555556</multiplier>
      <offset>-32</offset>
   </dataAxis>
</steadyStateResponse>
```

7.4.1.3.1.1 tml:phenAxis

Includes the attributes of hysteresisDirection, direct and units. The hysteresisDirection can be increasing, decreasing or both. The phenAxis is the axis along which the phenomenon is detected. The direct attribute is Boolean and stands for direction, which can be either true (forward) or false (backward). Units are the units along the axis.

7.4.1.3.1.1.1 tml:sequence

Sequence has a bindUID. A sequence contains enumerated values or sets of values of the transfer function, which are sampled from the transfer function plot.

7.4.1.3.1.1.2 tml:multiplier

Multiplier is a bindUID. The multiplier is applied to the transfer function sequence values.

7.4.1.3.1.1.3 tml:offset

Offset has a bindUID. The offset is applied to the transfer function sequence values, and is added to the specified sequence value multiplied by the multiplier.

7.4.1.3.1.1.4 tml:accuracy

The accuracy element includes the attributes of bindUID, type and factor. This specifies the accuracy of the transfer function and is used in uncertainty calculations.

7.4.1.3.1.1.5 tml:limits

This specifies limits on the transfer function. The limits tag includes the elements min, max, enumeration and stepSize. All of these elements are bindUID types.

7.4.1.3.1.1.6 tml:phenType

This is the type of phenomenon and is a string.

7.4.1.3.1.1.7 tml:calibration

This is the calibrated value of a calibrated source providing a calibrated stimulus to a sensor across it's TCF array, or it is the response from a data value stimulus for a calibrated transmitter or actuator across its TCF array. Typically it may just be a single value (e.g black body source). There may be multiple calibration points within the dynamic range of the transducer.

7.4.1.3.1.2 tml:dataAxis

Includes the attributes of hysteresisDirection and direct, as well as the elements of sequence, multiplier, offset, accuracy, limits and calibration. The dataAxis is the axis along which the data is sampled.

7.4.1.3.1.3 tml:continuity

The continuity can be discrete or continuous.

7.4.1.3.1.4 tml:interpolation

The interpolation can be undefined, returnToZero, lastValue or interpolate.

7.4.1.3.2 tml:impulseResponse

An impulse response describes the output of a transducer as a function of time given an input impulse of infinite magnitude and infinitesimal duration.

An impulse response function is a transfer function for characterizing the dynamic behavior of linear time invariant transducers. The impulse response is the response of a linear time invariant system to an impulse function. The convolution of the input signal with the impulse response function of a linear time invariant system will yield the output signal. Since linear time invariant systems are reversible, given the output signal, then the input signal can be derived through the reverse process. Figure 46 shows an example of a normalized impulse response.

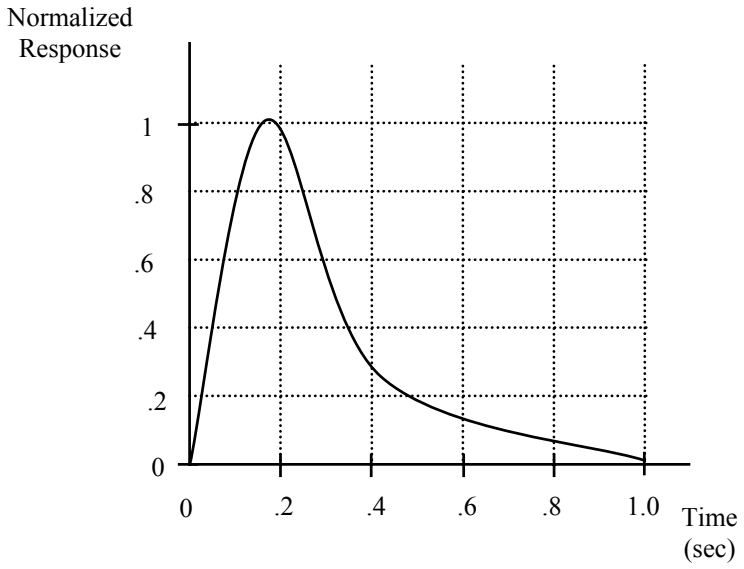


Figure 46 - Example of a normalized impulse response

The impulse response is useful for characterizing linear time invariant (LTI) transducers. If the transducer's response in known for a unit impulse input, then the transducer's response can be derived by convolving the input signal with the unit impulse response.

Example 1:

This example shows an impulse response with two axes, one amplitude and one time. Values are completely specified with an amplitude and a time axis quantity.

```
<impulseResponse>
   <amplitudeAxis>
      <sequence>0,0.5,1,0.75,0.5,0.25,0.125,0.031,0.015,0</sequence>
      <multiplier>1</multiplier>
      <offset>1</offset>
   </amplitudeAxis>
   <timeAxis>
      <sequence>0,0.5,1,1.5,2,3,4,5,6,7</sequence>
      <multiplier>1</multiplier>
      <offset>1</offset>
   </timeAxis>
</impulseResponse>
```


7.4.1.3.2.1 tml:amplitudeAxis

Includes the attributes of hysteresisDirection and direct, as well as the elements of sequence, multiplier, offset, accuracy and limits. The amplitudeAxis is the axis along which the amplitude of the impulse response is given.

7.4.1.3.2.2 tml:timeAxis

Includes the same attributes and elements as amplitudeAxis. The timeAxis is the axis along with the time of the impulse response is given.

7.4.1.3.3 tml:frequencyResponse

Includes the attribute of type.

A frequency response defines the transducer's ability to respond to various frequency inputs (modulated frequency) and the sensitive part of the frequency spectrum on which the transducer is responsive (carrier frequency).

The frequency response is a characterization of how a transducer responds to the various frequency components of a stimulus. TML uses the response amplitude versus frequency for three types of frequency related functions: carrier frequency response, modulated frequency response, and power spectral density.

Carrier frequency response represents the spectrum of carrier or frequencies which a sensor is sensitive to or is able to detect or respond to. TML expresses the functions as normalized curves, then applies gain and bias factors to the axis thus modeling the actual function. The frequency of some information is so low that the baseband signal of the information will not propagate through space. To communicate this information it can be carried by a higher frequency signal that will propagate through space. This higher frequency signal is called a carrier signal. The higher frequency carrier signal is modulated by the lower frequency information.

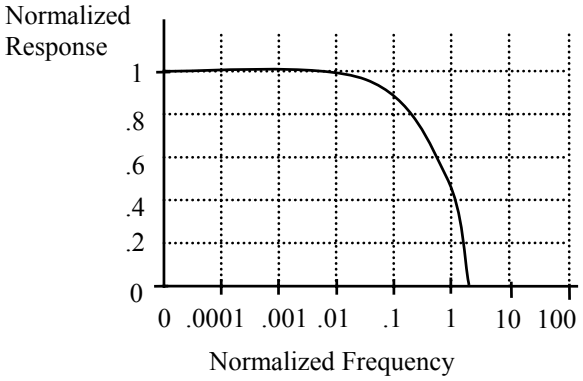


Figure 47 - Example of a low pass normalized frequency response

Modulation frequency response represents the spectrum of modulation frequencies which a transducer or process is sensitive to or is able to detect or respond to. TML expresses the functions as normalized curves, and then applies gain and bias factors to the axis to model the actual function. The modulation signal is a lower frequency signal that represents information that is put onto a higher frequency carrier signal.

Power Spectral Density refers to the spectrum of frequencies contained in a signal, baseband or modulated carrier. For transmitters this is the power spectrum of the emitted phenomenon. The spectral density of the wave, when multiplied by an appropriate factor, will give the power carried by the wave per unit frequency. This yields the power spectral density (PSD) of the signal. It is usually plotted as power –vs.- frequency. The units of spectral power density are commonly expressed in watts per hertz (W/Hz). TML expresses the functions as normalized curves then applies gain and bias factors to the axis thus modeling the actual function. (See carrier frequency response and modulation frequency response)

Example 1:

This example shows a simple frequency response with entries for amplitude, frequency and phase.

```
<frequencyResponse type="carrier">
   <amplitudeAxis>
      <sequence>1, 1, 0</sequence>
   </amplitudeAxis>
   <frequencyAxis>
      <sequence>0, 118, 120</sequence>
   </frequencyAxis>
   <phaseAxis>
      <sequence>0, 0, 0</sequence>
   </phaseAxis>
</frequencyResponse>

<frequencyResponse type="powerSpectralDensity">
   <powerAxis>
      <sequence>1, 1, 0</sequence>
   </powerAxis>
   <frequencyAxis>
      <sequence>0, 118, 120</sequence>
   </frequencyAxis>
</frequencyResponse>
```

The frequencyResponse tag includes the element tml:amplitudeAxis as well as the elements below.

7.4.1.3.3.1 tml:frequencyAxis

The frequencyAxis tag includes the same attributes and elements as amplitudeAxis. The frequencyAxis is the axis along which the frequency of the frequency response is given.

7.4.1.3.3.2 tml:phaseAxis

Includes the same attributes and elements as amplitudeAxis. The phaseAxis is the axis along which the phase of the frequency response is plotted.

7.4.1.3.3.3 tml:powerAxis

Includes the same attributes and elements as amplitudeAxis. The powerAxis is the axis along which the power spectral density is plotted for the frequency response.

7.4.1.3.4 tml:latencyTime

This captures latency time in seconds that is for all data captured from a transducer.

The latency time is the elapsed between when a phenonona changes and when that the data on the other side of the transducer reflects that change. There is only one latency value for a transducer, which correlates to the time it takes for the phenomona to propagate through a transducer, relative to the clk latch point. All other temporal measurements use the clk latch point as their reference point. Time should always be measured in the forward temporal direction, and as such for a sensor, the latency time is most likely negative. **Figure 48** illustrates the time definitions.

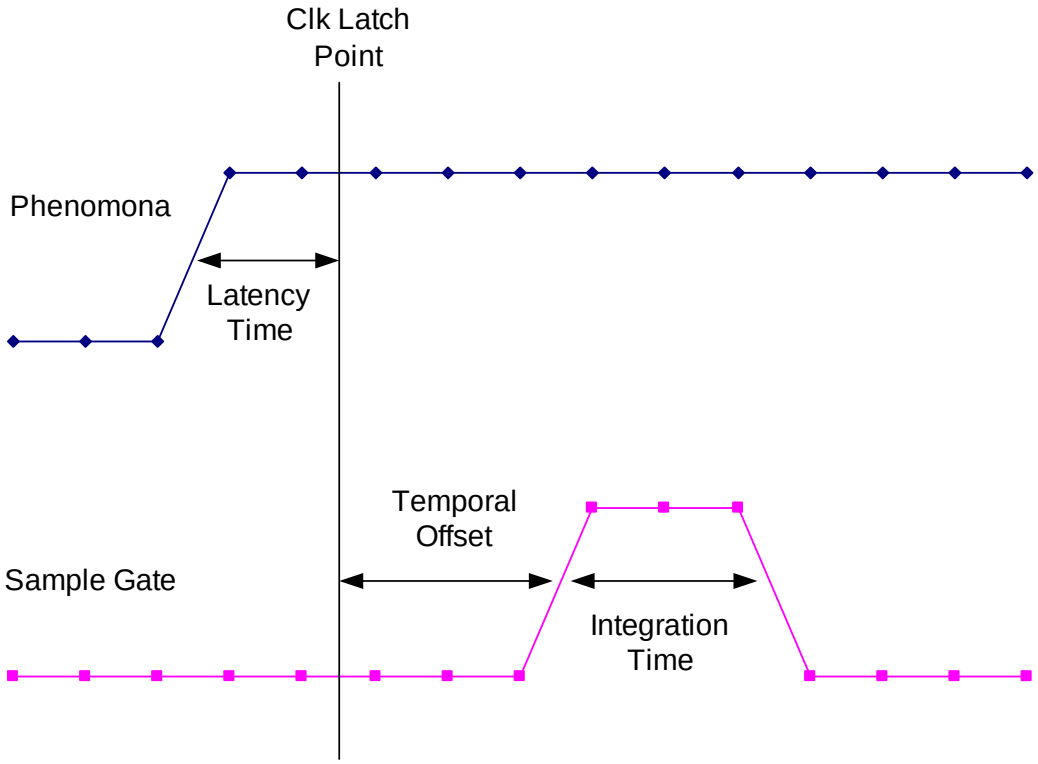


Figure 48 - Time Definitions

The latencyTime tag includes the elements of value and accuracy.

7.4.1.3.5 tml:temporalOffset

The temporal offset is the elapsed time from the "clk latch point" to when a specific unit in a transducer logical data model starts to process information.

7.4.1.3.5.1 tml:absoluteTime

Includes the attributes of axis and index. The absoluteTime is the absolute time during which a measurement occurred. Axis is the axis that the absolute time refers to. Index is the index number along the specified axis.

7.4.1.3.5.1.1 tml:timeValue

Includes the elements value and accuracy. The timeValue is a listing of time values indicating temporal offset.

7.4.1.3.5.1.2 tml:valueAxis

Includes the attributes of axis and index, as well as the elements timeValue and valueAxis. The valueAxis specifies the axis along which the temporal offsets occur.

7.4.1.3.5.2 tml:relativeTime

Includes the attributes of axis and index, as well as the elements of timeValue and valueAxis and the elements below. The relativeTime is the time of individual measurements relative to the clock latch point.

7.4.1.3.5.2.1 tml:referencePeriodLength

The referencePeriodLength has a bindUID. The referencePeriodLength is the period length to which the relative time is given.

7.4.1.3.5.2.2 tml:periodLength

The periodLength has a bindUID. The periodLength is the relative period length to which the relative time is given.

7.4.1.3.6 tml:integrationTime

The integration time is the elapsed time during which the transducer is processing information. For a sensor it is the time during which the sensor actually samples the phenomona. During this time any modulation of the phenomena is not uniquely distinguishable.

The integration time is the time required by a receiver to accumulate a reading or the time for a transmitter to apply an action. Time resolution can be determined no finer than the ambiguity time. For example, when a camera shutter opens for 1/30 of a second that is the integration time for that set of measurements. When an aircraft flies, an air sampler opens the sampling instrument at 3pm and closes it at 5pm, then 2 hours is the integration time for that sensor. Any modulation that occurs within the integration time cannot be determined. One value corresponds to response or stimulus depending on if the transducer is a receiver or transmitter respectively.

The integrationTime tag includes the elements absoluteTime and relativeTime.

7.4.1.3.7 tml:noiseFigure

Includes the element value, which includes in turn the elements value and valueAxis.

The noise figure of a transducer or a process is the ratio of the output signal to noise ratio divided by the output signal to noise ratio. The noise figure is used to indicate how much noise a transducer or a process adds in transforming the input to the output.

7.4.1.3.8 tml:spatialGeometry

The spatialGeometry tag includes the attribute of ambShapeType. The ambShapeType can be ifoi, processedShape or externalSurface.

The geometry model describes the mapping of data to real world spatial coordinates or ambiguities whichever the case may be. The real world geometry relations of data are described using interior or relative geometry, and exterior or absolute geometry. A geometry model is a collection of values that describe the relative space time relationships of each data unit, in a transducer TCF array, relative to a coordinate reference systems such that an interior geometry can be spatial and temporally invariant.

Interior geometry can be described such that it is independent of the location, attitude, date, and time. Once the interior geometry is defined then it can be positioned relative to absolute spatial and temporal datums.

This element defines the geometry model for each sample or measure of the transducer. This is required to map data to its enterprise spatial location.

Remote transducers have this characteristic. In-situ transducers do not have an ambiguity space. A data unit corresponds to a phenomena, the phenomena may be localized (i.e. come for a specific point or the phenomena may be distributed (accumulation from a field) the ambiguity space is the volume of space of which the phenomena is know to exist. An IFOI is one type of ambiguity space. An ambiguity space will have a shape and a position defined relative to a specific spatial reference system. In some situations, the ambiguity space may be a single point. In other cases, the shape will define some volume of space (line, plane, sphere, arc, cone, cardoid, teardrop, etc.). The resultant ambiguity space can be composed of multiple ambiguity spaces of either solid, inverse solid, or thin surfaces. The figures below shows some examples of ambiguity space.

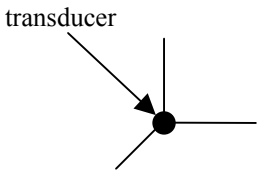


Figure 49 - In-situ transducer (no ambiguity space)

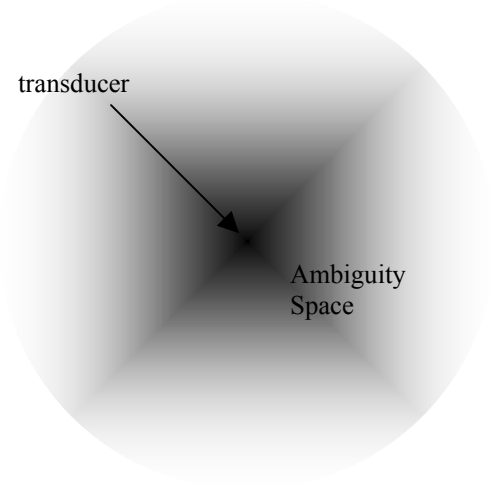


Figure 50 - Remote transducer omni-directional ambiguity space

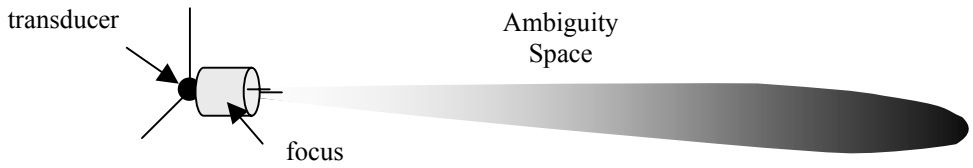


Figure 51 - Remote transducer directional ambiguity space

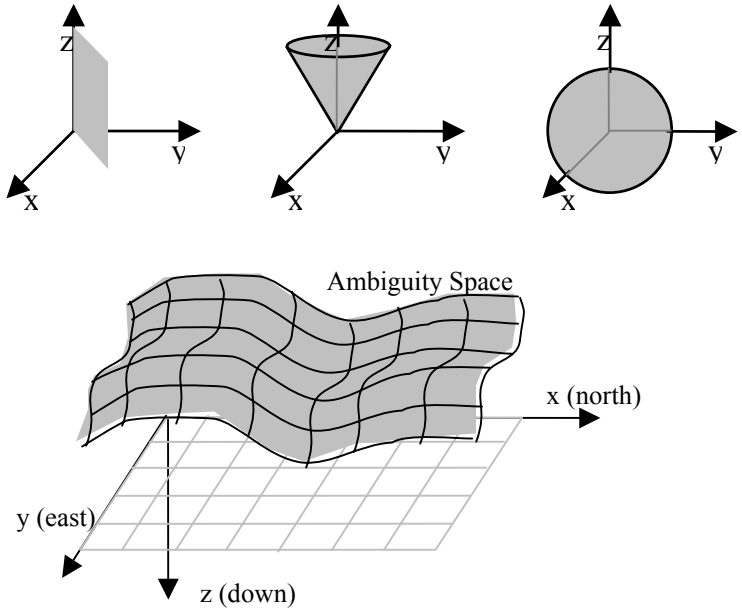


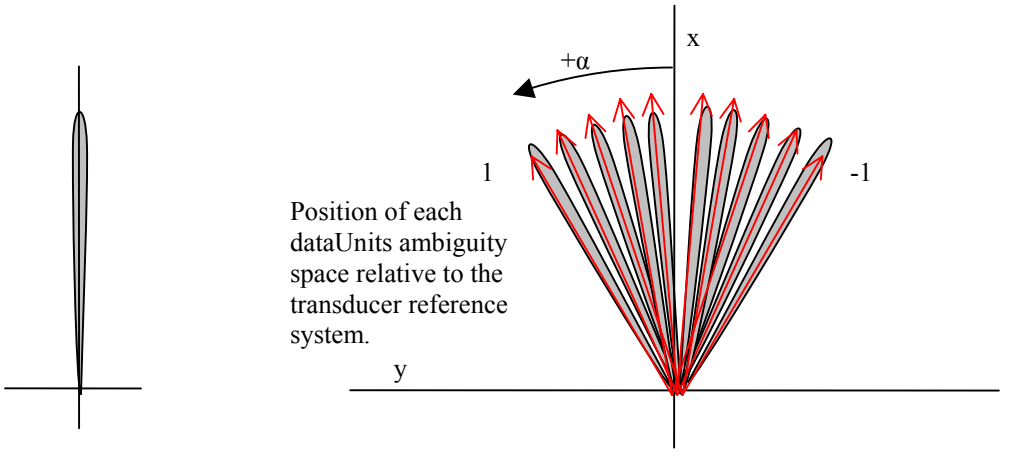
Figure 52 - Examples of geometric and irregularly shaped ambiguity space

A primary capability of TML is to facilitate the precise and accurate geo-positioning of transducer data. This enables the rapid space and time association of transducer data from disparate transducers as well as with other geospatial data (e.g. maps, DTED). To this end, TML goes to great lengths to maintain relative and absolute spatial and temporal measurements for each transducer data unit, as well as describing the logical relationships of multiple transducers in a multi transducer system.

There exists a spatial extent where we believe, to a certain probability, that an object (or a property of an object) exist, that spatial extent is called an ambiguity space. For example, if we know that an object will exist on the earth surface, then the earth surface is an ambiguity space. If we know an object exist at a certain range and within a certain remote transducer field the intersection of the two ambiguity spaces results in a smaller ambiguity space.

Objects known to exist within an ambiguity space can be more precisely located by intersecting other known ambiguity spaces for that object.

To improve the sensitivity or improve the effective power density or improve spatial resolution a focused field of energy or received area may be described. This is ambiguity space. The ambiguity space is a space where it is highly probable that the object of feature creating the phenomenon is located in. For example, an Instantaneous Field of View of a camera pixel is an ambiguity space for that camera. The antenna pattern for a radar is an ambiguity space for that radar. There are energy field ambiguities refereed to in TML as Instantaneous Field of Influence (IFOI). Transducer data may have multiple ambiguity spaces associated with it. For example after radar data is received and it is associated with the transmit pulse, each dataUnit in the received range gate has a unique rang associated with it. So we will be able to determine that objects within the transmit and receive antenna pattern and at a constant range (sphere) could contribute to the response. The ambiguity shape may take on a variety of shapes. The shapes may be well formed like the ones shown in Figure 53 or they may be irregularly shaped such as a terrain model. Ambiguity shapes may be described using any of the coordinates available in the three coordinates systems available in TML (rectangular, spherical, and cylindrical) relative to any of the reference systems (ECEF, Local, Transducer).



One TCF of 10 dataSets

data index number	1	2	3	4	5	6	7	8	9	10
dataset α angle	-1	-.7	-.5	-.3	-.1	.1	.3	.5	.7	1
dataset β angle	0	0	0	0	0	0	0	0	0	0
dataSet time	0	1	2	3	4	5	6	7	8	9

Figure 53 - Distribution of ambiguity shapes relative to a transducer reference frame

Ambiguity shapes can be modeled with multiple 2-dimensional functions or a 3-dimensional function. The ambiguity shape can represent a surface a closed or solid shape or an open or inverse solid. This enables multiple ambiguity shapes to intersect to determine phenomenon spatial constraints more accurately.

The internal spatial characteristics of transducer data will be described by characterizing the ambiguity shape and position for each data unit. Typically, the ambiguity shape will be basically the same for every data unit in the TCF. There may be small variances over the array, which can be described by an array variable. However, the position of the ambiguity shape for each data unit is different. Figure 53 above illustrates the characterization of the internal geometry relative to the transducer reference frame and the start of the scan. In this case, the spatial position of the ambiguity shape is described relative to a transducer reference system and the temporal reference is described relative to the start of the scan. The alpha and beta angles define the orientation of each of the ambiguity shapes relative to the transducer reference system. If the reference system is defined properly the positional variances of each of the data units may change only one or two coordinate parameter out of the six possible (3 location and 3 attitude). For example, in this case the scan was in the direction of constant beta. Only one coordinate varied, alpha.

The spatialGeometry tag includes an identification element as well as the elements below.

7.4.1.3.8.1 tml:ambShape

Includes the attributes of shapeType, id and ref, as well as the name and ambShape elements and the elements below. The ambShape is the shape of the ambiguity space. The shapeType can be solid, invSolid, or surface.

7.4.1.3.8.1.1 tml:probability

The probability has a bindUID. The probability describes the uncertainty of the ambiguity shape.

7.4.1.3.8.1.2 tml:pointCloudList

The pointCloudList tag includes the attribute of coordSystem. The pointCloudList is a list of points in the point cloud. The attribute coordSystem can be spherical or rectangular.

7.4.1.3.8.1.2.1 tml:pointCoordAxis

Includes the attribute of coordName and the elements of sequence, multiplier, offset and accuracy. The pointCoordAxis is the axis for the point coordinates. CoordName is the name of the coordinate, like x, y, z, alpha, beta or rho.

7.4.1.3.8.2 tml:ambShapeFramePosition

Includes the attributes of coordSystem, relToRefSys and UIDref. The ambShapeFramePosition gives the positions of the coordinate systems. The attribute

relToRefSys can be earth, transducer or localEarth. UIDref is the UID reference given to this coord system.

7.4.1.3.8.2.1 tml:posCoordAxis

Includes the attributes of ambShapeRef, index, axis and coordName, as well as the elements value and valueAxis. The posCoordAxis is the position coordinate axis which are the axes that define the coordinate system for the ambiguity shapes. The ambShapeRef attribute is the reference for the ambiguity shape for which this is being defined. The coordName can be x, y, z, alpha, beta, rho, omega, phi, kappa.

7.4.1.3.8.3 tml:transAperSize

Includes the attribute of coordSystem. The transAperSize is the aperture size of the transducer if it is a camera for example.

7.4.1.3.8.3.1 tml:crossSectionArea

Includes the elements sequence, multiplier, offset and accuracy. The crossSectionArea defines the cross sectional area of the aperture.

7.4.1.3.8.3.2 tml:pointCoordAxis

Includes the attribute of coordName, as well as the elements of sequence, multiplier, offset and accuracy. The pointCoordAxis are the axes that define points for the aperture cross section.

7.4.1.3.9 tml:other

The other tag can be of any type. The other element is a placeholder for other behavioral models, such as Replacement Sensor Model (RSM) or Rational Polynomial Coefficients (RPC).

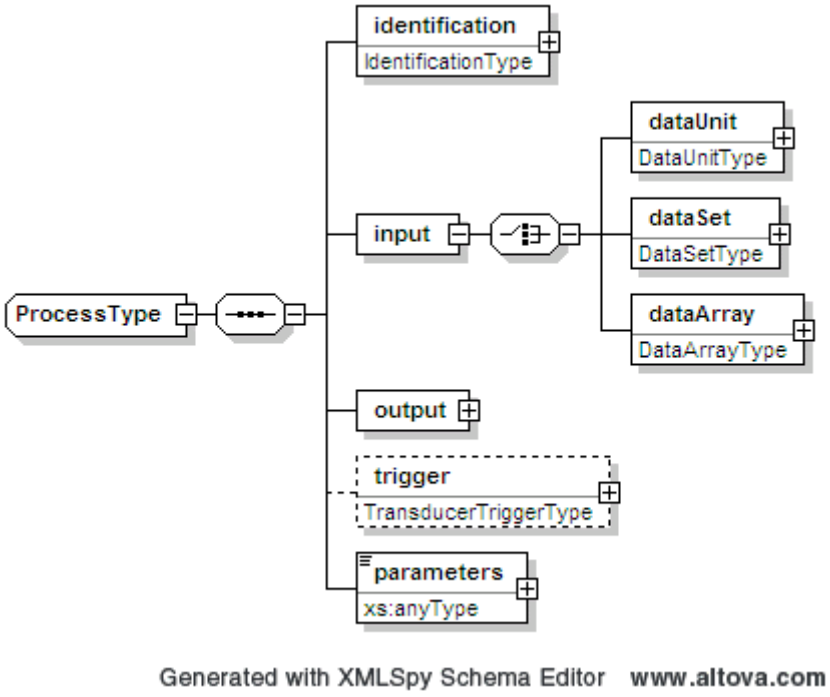


Figure 54. TML Process Diagram

7.5 TML Process

The tml:processList tag includes the elements described below.

7.5.1 tml:process

Includes the attributes of id and ref, as well as the identification and trigger elements and the elements below.

A process is a function that takes one or more digital inputs, and based on parameters and methodologies, produces a digital output. A process can receive data inputs and/or output data to transducers or other processes.

A TML process system is a process which inputs and/or outputs TML data. A process system does not contain transducers. The input and output to a process system is data. One or both of the data forms must be TML data. A process algorithm is not a TML system process until either the input and/or output is in TML. Process systems can process TML data prior to being transmitted by a transmitter or after being received by a receiver.

7.5.1.1 tml:input

Includes the elements dataUnit, dataSet and dataArray. The input can be one or more inputs into the process, and can be defined as units, sets or arrays.

7.5.1.2 tml:output

Includes the elements dataUnit, dataSet and dataArray. The output can be one or more outputs defined as units, sets, or arrays.

7.5.1.3 tml:parameters

The parameters can be of any type. The parameters are any parameters that help define the process.

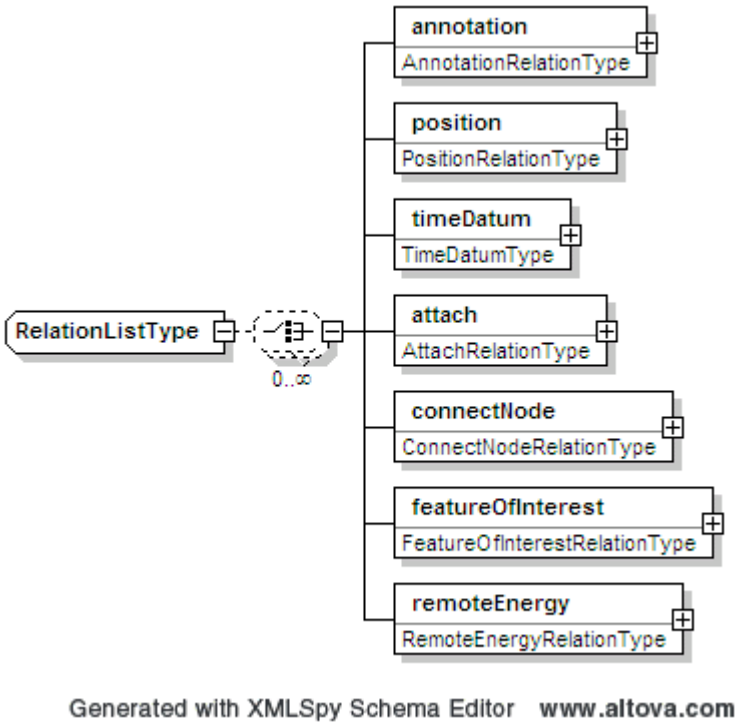


Figure 55. TML Relation Diagram

7.6 TML Relations

The tml:relationList tag includes the elements below. Relations describe the logical and physical relations between the transducers in a system, and between systems.

7.6.1 tml:annotation

Includes the attribute of UIDref. The annotation is a comment that can describe relationships.

7.6.1.1 tml:documentation

Includes the attribute of source and UIDref. Documentation is a place to put a note.

7.6.2 tml:position

Includes the attributes of transducerUIDref, coordSystem, relativeRefSystem and relToCompUIDref. The transducerUIDref is the reference UID of the transducer. The coordSystem can be wgs84geodetic, spherical, or rectangular. The relativeRefSystem is earth, local or transducer. The relToCompUIDref is the reference of the relative component.

The position of an object (feature or transducer) is determined by both the location and attitude relative to some spatial reference system (e.g. earth). It is described using a spatial coordinate system (e.g. spherical), with location coordinates (e.g. α, β, ρ) and attitude coordinates (e.g. ω, φ, κ) from that system.

7.6.2.1 tml:posCoord

Includes the attributes of coordName and uom, as well as the elements value and accuracy. This defines the position coordinate system for the relations. The coordName gives the name of the coordinate, the uom defines the unit of measure for that coordinate axis, the value element contains values along the coordinate axis and the accuracy gives the accuracy of those values.

7.6.2.2 tml:velCoord

Includes the attributes of coordName and uom, as well as the elements value and accuracy. This defines the velocity coordinate system for the relations. The coordName gives the name of the coordinate, the uom defines the unit of measure for that coordinate axis, the value element contains values along the coordinate axis and the accuracy gives the accuracy of those values.

7.6.2.3 tml:accelCoord

Includes the attributes of coordName and uom, as well as the elements value and accuracy. This defines the acceleration coordinate system for the relations. The coordName gives the name of the coordinate, the uom defines the unit of measure for that coordinate axis, the value element contains values along the coordinate axis and the accuracy gives the accuracy of those values.

7.6.3 tml:timeDatum

Includes the attributes of sysClkRef, timeDatum and ref. This gives the time information for the relations. The sysclkRef is a reference to the system clock to use, the timeDatum is a string giving the time reference system which is currently limited to UTC, and the ref is a reference identifier for this particular time datum.

7.6.3.1 tml:timestamp

Includes the attribute of bindUIDref. This is the time stamp value.

7.6.3.2 tml:year

Includes the attribute of bindUIDref. This contains the year of the time stamp.

7.6.3.3 tml:month

Includes the attribute of bindUIDref. This contains the month of the time stamp.

7.6.3.4 tml:day

Includes the attribute of bindUIDref. This contains the day of the time stamp.

7.6.3.5 tml:hour

Includes the attribute of bindUIDref. This contains the hour of the time stamp.

7.6.3.6 tml:minute

Includes the attribute of bindUIDref. This contains the minute of the time stamp.

7.6.3.7 tml:second

Includes the attribute of bindUIDref. This contains the second of the time stamp.

7.6.4 tml:attach

The attach tag is a bindUIDref. It includes the element value as well as the element below. This gives the reference of the attached transducer. This element is used to describe relationships of data values for system metadata which originate external to a system or transducer and are known after the transducer or system description have been built or are contained in the live data stream.

7.6.4.1 tml:attachDesc

This contains a description of the attached transducer.

7.6.5 tml:connectNode

This contains information about the node to which the connection is made.

7.6.5.1 tml:connectDescription

This is a description of the connection node.

7.6.5.2 tml:connectUID

Includes the attribute of bindUIDref. This is the UID reference of the node to which the connection is made.

7.6.6 tml:featureOfInterest

Includes the attribute of bindUIDRef.

This is the object, subject or feature that the phenomenon is affecting (transmitter case) or the phenomena property that was affected from the feature (receiver case) to which the data (identified in the dataUIDref of this element) relates. These objects are real world physical objects or features to which the transducer or dataUnit identified in the attribute relates. The object may be displaced from the transducer (remote) or the transducer (transducing element) may be in contact with the object directly (in-situ). The UIDref attribute of the element identifies the dataUnit, dataGroup, or dataSet UID to which this features corresponds. If the feature has a UID, it should be referenced in the child element identification - UID element. For example, the feature of interest may be a component within the system (transducer, physical component, process etc.). If the feature does not have a UID then just a description would be adequate. Sometimes the object is not known until the data is interpreted by a process. If the feature identification is provided by some other process, then the UIDref of the description or UID elements points to the transducer or process providing the data (feature Of Interest descriptions) which corresponds to the dataUnit or dataSet which is identified in the attribute of this element. The data structure of the process providing the FOI description will be the same as the data structure of the source transducer data (i.e. may be a data array).

Features are physical entities (of matter or energy) that can be observed or influenced through transducers. Features exhibit properties, which are made evident through observable phenomenon properties. A phenomenon can also be made to change or revise properties of features.

Table 8 - Example features

Examples of Features	
– Earth surface – Geographic Features – Boundaries & Regions – Shoreline, political – Transportation – Roads, railroads – Structure – Cities, bridges, buildings – Communication & Utilities	– Earth atmosphere – Troposphere – Stratosphere – Mesosphere – Thermosphere – Exosphere – Ionosphere – Vehicles – Land

– Electric, gas – Land Surfaces – Vegetation – Crops – Forest – Hydrological features – Oceans, lakes, rivers, streams – Elevation	– Air – Space – Water Surface – Underwater – Vehicle Payloads – Transducers – Weapons – Cargo – People

7.6.6.1 tml:identification

This element is different from the normal identification tag type because it only includes the element description. It includes the attribute of UIDref.

7.6.6.1.1 tml:description

This is a description of the feature of interest in text.

7.6.7 tml:remoteEnergy

Includes the attributes of UIDRef and type. Remote energy refers to energy sources or transmissions which are separated from the transducer. As such, remote energy has an effect on local transducer behavior, but can be influenced by the medium through which the energy propagates. The UIDRef refers to the transducer sensing or transmitting the remote energy, and the type gives the type of remote behavior (reflected, emitted, both or unknown).

7.6.7.1 tml:medium

This describes the medium through which the remote energy is propagated. This tag has an attribute, propagationMechanism, which can take on values radiation, conduction, convection or osmosis.

7.6.7.2 tml:reflEnergySource

This describes the reflected energy source and is a text string. This element has two attributes, energySourceType and reflPhenSource. energySourceType can take on values natural or artificial, while reflPhenSource is any URI which refers to the reflected phenomenon source.

Annex A (informative) - Basic Data Types

Basic Types (Informative)

There are several basic data types in use throughout this specification, and they are defined in this section.

A.1 BindType

Schema:

```
<complexType name="BindType">
   <simpleContent>
      <extension base="anySimpleType">
         <attribute name="bindUID" type="xs:Name"/>
      </extension>
   </simpleContent>
</complexType>
```

BindType provides a bindable field that can be used to fill in a value directly, or provide a default value with a way to link it to something else later.

Example:

```
<element name="value" type="BindType"/>

<value>5</value>
<value bindUID="SOMEUID"/>
<value bindUID="SOMEUID">5</value>
```

First example you use the value of 5 directly. Second example lookup the changing value elsewhere. Third example you use the default value of 5 and then lookup the changing value elsewhere for the current value.

[The xs:Name is used for bindUID and for id and ref attributes throughout. This is used throughout the document for ID, UID and reference attributes.
http://www.w3.org/TR/xmlschema-2/#Name]

A.2 BindRefType

Schema:

```
<complexType name="BindRefType">
   <simpleContent>
      <extension base="anySimpleType">
         <attribute name="bindUIDref" type="xs:Name"/>
```



```
      </extension>
   </simpleContent>
</complexType>
```

BindRefType provides a field that can be linked to another bind Point. It behaves the same as a BindType but it requires an existing BindPoint to be used.

Example:

```
<element name="value" type="BindRefType"/>

<value>5</value>
<value bindUIDref="SOMEUID"/>
<value bindUIDref="SOMEUID">5</value>
```

In the first example the value of 5 is used directly. In the second example the changing value is looked up elsewhere. In the third example the default value of 5 is used and the changing value is then looked up elsewhere for the current value.

A.3 AccuracyType

Schema:

```
<complexType name="AccuracyType">
   <simpleContent>
      <extension base="BindType">
         <attribute name="type" use="required">
            <simpleType>
               <restriction base="string">
                  <enumeration value="relative"/>
                  <enumeration value="absolute"/>
               </restriction>
            </simpleType>
         </attribute>
         <attribute name="factor">
            <simpleType>
               <restriction base="string">
                  <enumeration value="1Sigma"/>
                  <enumeration value="2Sigma"/>
                  <enumeration value="3Sigma"/>
                  <enumeration value="4Sigma"/>
                  <enumeration value="5Sigma"/>
                  <enumeration value="6Sigma"/>
                  <enumeration value="percent"/>
                  <enumeration value="range"/>
               </restriction>
            </simpleType>
         </attribute>
      </extension>
   </simpleContent>
</complexType>
```

This type provided the basic accuracy element type that is used throughout TML when accuracy is to be specified.

Example:
```
<element name="accuracy" type=" AccuracyType "/>

<accuracy type="relative">0.5</accuracy>
<accuracy type="absolute" factor="3Sigma">1.0</accuracy>
```

In the first example the value of 5 is used directly. In the second example the changing value is looked up elsewhere. In the third example the default value of 5 is used and the changing value is looked up elsewhere for the current value.

A.4 ValueAccuracyType

Schema:
```
<complexType name="ValueAccuracyType">
   <sequence>
      <element name="value" type="BindType"/>
      <element name="accuracy" type="AccuracyType" minOccurs="0"
maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

This provides a bindable element for specifying a value along with accuracy information for the value specified.

Example:
```
<element name="period" type="ValueAccuracyType"/>
<period type="relative">0.5
    <value>5</value>
    <accuracy type="relative">0.5</accuracy>
    <accuracy type="absolute" factor="3Sigma">1.0</accuracy>
</period>
```

A.5 ValueAxisType

```
<xs:complexType name="ValueAxisType">
    <xs:choice>
        <xs:element name="value" type="BindType"/>
        <xs:element name="valueAxis" type="ValueAxisType"/>
    </xs:choice>
    <xs:attribute name="axis"/>
    <xs:attribute name="index"/>
</xs:complexType>
```

This provides a mechanism to assign a value for every unit in a Logical Data Model or just specific sub groups.

Annex B
(normative)

Abstract test suite

B.1 General

A validation and verification service will confirm the proper implementation of TML messages. A description of the test suite will be described in latter versions of TML.

Annex C
(normative)

XML Schema Documents

In addition to this document, this specification includes several normative XML Schema Documents. These XML Schema Documents are bundled in a zip file with the present document. After OGC acceptance of a Version 1.0.0 of this specification, these XML Schema Documents will also be posted online at the URL http://schemas.opengeospatial.net/TML/1.0.0. In the event of a discrepancy between the bundled and online versions of the XML Schema Documents, this document shall be considered authoritative.

Annex D
(normative)

TML Schema

```
xmlns="http://www.opengis.org/tml"
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by Jon Schlueter
(IRIS Corporation) -->
<xs:schema xmlns="http://www.opengis.org/tml"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.opengis.org/tml" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="tml">
        <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element name="systemList" type="DataSystemListType"/>
                <xs:element name="transducerList" type="TransducerListType"/>
                <xs:element name="processList" type="ProcessListType"/>
                <xs:element name="relationList" type="RelationListType"/>
                <xs:element name="clusterDescList" type="ClusterDescListType"/>
                <xs:element name="data" type="DataClusterType"/>
            </xs:choice>
            <xs:attribute name="version" type="xs:string" use="required"
fixed="1.0.1 20060301"/>
            <xs:attribute name="uid" type="xs:anyURI" use="optional"/>
        </xs:complexType>
    </xs:element>
    <!--Base Types-->
    <xs:complexType name="BindType">
        <xs:simpleContent>
            <xs:extension base="xs:anySimpleType">
                <xs:attribute name="bindUID" type="xs:Name"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="BindRefType">
        <xs:simpleContent>
            <xs:extension base="xs:anySimpleType">
                <xs:attribute name="bindUIDref" type="xs:Name"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="AccuracyType">
        <xs:simpleContent>
            <xs:extension base="BindType">
                <xs:attribute name="type" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="relative"/>
                            <xs:enumeration value="absolute"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="factor">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="1Sigma"/>
                            <xs:enumeration value="2Sigma"/>
```



```
                        <xs:enumeration value="3Sigma"/>
                        <xs:enumeration value="4Sigma"/>
                        <xs:enumeration value="5Sigma"/>
                        <xs:enumeration value="6Sigma"/>
                        <xs:enumeration value="percent"/>
                        <xs:enumeration value="range"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="ValueAccuracyType">
    <xs:sequence>
        <xs:element name="value" type="BindType"/>
        <xs:element name="accuracy" type="AccuracyType" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="TransferFunctionAxisType">
    <xs:sequence minOccurs="0">
        <xs:element name="sequence" type="BindType" minOccurs="0"/>
        <xs:element name="multiplier" type="BindType" minOccurs="0"/>
        <xs:element name="offset" type="BindType" minOccurs="0"/>
        <xs:element name="accuracy" type="AccuracyType" minOccurs="0"
maxOccurs="unbounded"/>
        <xs:element name="limits" minOccurs="0">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="min" type="BindType" minOccurs="0"/>
                    <xs:element name="max" type="BindType" minOccurs="0"/>
                    <xs:element name="enumeration" type="BindType"
minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="stepsize" type="BindType"
minOccurs="0"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="hysteresisDirection">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="both"/>
                <xs:enumeration value="increasing"/>
                <xs:enumeration value="decreasing"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="direct" type="xs:boolean"/>
</xs:complexType>
<xs:complexType name="ValueAxisType">
    <xs:choice>
        <xs:element name="value" type="BindType"/>
        <xs:element name="valueAxis" type="ValueAxisType"
maxOccurs="unbounded"/>
    </xs:choice>
    <xs:attribute name="axis"/>
    <xs:attribute name="index"/>
</xs:complexType>
<!--List Types-->
<xs:complexType name="DataSystemListType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="system" type="DataSystemType"/>
    </xs:choice>
</xs:complexType>
```



```
    <xs:complexType name="TransducerListType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="transducer" type="TransducerType"/>
        </xs:choice>
    </xs:complexType>
    <xs:complexType name="ProcessListType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="process" type="ProcessType"/>
        </xs:choice>
    </xs:complexType>
    <xs:complexType name="RelationListType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="annotation" type="AnnotationRelationType"/>
            <xs:element name="position" type="PositionRelationType"/>
            <xs:element name="timeDatum" type="TimeDatumType"/>
            <xs:element name="attach" type="AttachRelationType"/>
            <xs:element name="connectNode" type="ConnectNodeRelationType"/>
            <xs:element name="featureOfInterest"
type="FeatureOfInterestRelationType"/>
            <xs:element name="remoteEnergy" type="RemoteEnergyRelationType"/>
        </xs:choice>
    </xs:complexType>
    <xs:complexType name="ClusterDescListType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="clusterDesc" type="DataClusterDescType"/>
        </xs:choice>
    </xs:complexType>
    <!--Element Types-->
    <xs:complexType name="DataSystemType">
        <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="identification" type="IdentificationType"/>
            <xs:element name="sysClk" type="SysClkType"/>
            <xs:element name="systemList" type="DataSystemListType"/>
            <xs:element name="transducerList" type="TransducerListType"/>
            <xs:element name="processList" type="ProcessListType"/>
            <xs:element name="relationList" type="RelationListType"/>
        </xs:choice>
        <xs:attribute name="id" type="xs:Name"/>
        <xs:attribute name="ref" type="xs:Name"/>
    </xs:complexType>
    <xs:complexType name="TransducerType">
        <xs:sequence minOccurs="0">
            <xs:element name="identification" type="IdentificationType"/>
            <xs:element name="trigger" type="TransducerTriggerType"
minOccurs="0"/>
            <xs:element name="logicalDataModel" type="LogicalDataModelType"
minOccurs="0"/>
            <xs:element name="behaviorModels" type="BehaviorModelsType"
minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:Name"/>
        <xs:attribute name="ref" type="xs:Name"/>
    </xs:complexType>
    <xs:complexType name="ProcessType">
        <xs:sequence>
            <xs:element name="identification" type="IdentificationType"/>
            <xs:element name="input">
                <xs:complexType>
                    <xs:choice>
                        <xs:element name="dataUnit" type="DataUnitType"/>
                        <xs:element name="dataSet" type="DataSetType"/>
                        <xs:element name="dataArray" type="DataArrayType"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
```



```
            <xs:element name="output">
                <xs:complexType>
                    <xs:choice>
                        <xs:element name="dataUnit" type="DataUnitType"/>
                        <xs:element name="dataSet" type="DataSetType"/>
                        <xs:element name="dataArray" type="DataArrayType"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <xs:element name="trigger" type="TransducerTriggerType"
minOccurs="0"/>
            <xs:element name="parameters" type="xs:anyType"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:Name"/>
        <xs:attribute name="ref" type="xs:Name"/>
    </xs:complexType>
    <xs:complexType name="DataClusterDescType">
        <xs:sequence minOccurs="0">
            <xs:element name="identification" type="IdentificationType"
minOccurs="0"/>
            <xs:element name="encoding" minOccurs="0">
                <xs:complexType>
                    <xs:choice>
                        <xs:element name="encodeUnit" type="EncodeUnitType"/>
                        <xs:element name="encodeSet" type="EncodeSetType"/>
                        <xs:element name="encodeArray" type="EncodeArrayType"/>
                    </xs:choice>
                    <xs:attribute name="encodingType">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:enumeration value="binary_base64"/>
                                <xs:enumeration value="binary_base16"/>
                                <xs:enumeration value="text_delimited"/>
                                <xs:enumeration value="text_fixed_width"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:attribute>
                    <xs:attribute name="complexity" type="xs:integer"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="id" type="xs:Name"/>
        <xs:attribute name="ref" type="xs:Name"/>
    </xs:complexType>
    <xs:complexType name="DataClusterType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="ref" type="xs:Name"/>
                <xs:attribute name="clk" type="xs:integer"/>
                <xs:attribute name="dateTime" type="xs:date"/>
                <xs:attribute name="reference" type="xs:anyURI"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--Support Types-->
    <xs:complexType name="IdentificationType">
        <xs:sequence>
            <xs:element name="UID" type="BindType" minOccurs="0"/>
            <xs:element name="name" type="BindType" minOccurs="0"/>
            <xs:element name="description" type="BindType" minOccurs="0"/>
            <xs:element name="type" type="BindType" minOccurs="0"/>
            <xs:element name="lastModification" type="BindType" minOccurs="0"/>
            <xs:element name="serialNumber" type="BindType" minOccurs="0"/>
            <xs:element name="modelNumber" type="BindType" minOccurs="0"/>
        </xs:sequence>
```



```
    </xs:complexType>
    <!--System Entry Types-->
    <xs:complexType name="SysClkType">
        <xs:sequence>
            <xs:element name="identification" type="IdentificationType"
minOccurs="0"/>
            <xs:choice minOccurs="0">
                <xs:element name="period" type="ValueAccuracyType"
minOccurs="0"/>
                <xs:element name="frequency" type="ValueAccuracyType"
minOccurs="0"/>
            </xs:choice>
            <xs:element name="minCount" type="BindType" minOccurs="0"/>
            <xs:element name="maxCount" type="BindType" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:Name"/>
        <xs:attribute name="ref" type="xs:Name"/>
    </xs:complexType>
    <!--Transducer Entry Types-->
    <xs:complexType name="TransducerTriggerType">
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="externalReference" type="BindType"/>
            <xs:element name="triggerMode" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="BindType">
                            <xs:attribute name="triggerType">
                                <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                        <xs:enumeration value="risingEdge"/>
                                        <xs:enumeration value="fallingEdge"/>
                                        <xs:enumeration value="rollover"/>
                                        <xs:enumeration value="match"/>
                                        <xs:enumeration value="modulus"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:attribute>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="LogicalDataModelType">
        <xs:sequence minOccurs="0">
            <xs:element name="transducertype" minOccurs="0">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="insitu"/>
                        <xs:enumeration value="remote"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="invariance" minOccurs="0">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="PositionalInvariant"/>
                        <xs:enumeration value="RotationalInvariant"/>
                        <xs:enumeration value="PositionalRotationalInvariant"/>
                        <xs:enumeration value="None"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:choice>
```



```
            <xs:element name="dataUnit" type="DataUnitType"/>
            <xs:element name="dataSet" type="DataSetType"/>
            <xs:element name="dataArray" type="DataArrayType"/>
        </xs:choice>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="DataArrayType">
    <xs:sequence>
        <xs:element name="name" type="xs:string" minOccurs="0"/>
        <xs:element name="arraySize" type="xs:integer"/>
        <xs:element name="indexSequence" type="BindType" minOccurs="0"/>
        <xs:choice>
            <xs:element name="dataUnit" type="DataUnitType"/>
            <xs:element name="dataSet" type="DataSetType"/>
            <xs:element name="dataArray" type="DataArrayType"/>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="id" type="xs:Name"/>
    <xs:attribute name="ref" type="xs:Name"/>
</xs:complexType>
<xs:complexType name="DataSetType">
    <xs:sequence>
        <xs:element name="name" type="xs:string" minOccurs="0"/>
        <xs:element name="indexSequence" type="BindType" minOccurs="0"/>
        <xs:choice maxOccurs="unbounded">
            <xs:element name="dataUnit" type="DataUnitType"/>
            <xs:element name="dataSet" type="DataSetType"/>
            <xs:element name="dataArray" type="DataArrayType"/>
        </xs:choice>
    </xs:sequence>
    <xs:attribute name="id" type="xs:Name"/>
    <xs:attribute name="ref" type="xs:Name"/>
</xs:complexType>
<xs:complexType name="DataUnitType">
    <xs:sequence>
        <xs:element name="name" type="xs:string" minOccurs="0"/>
        <xs:element name="indexSequence" type="BindType" minOccurs="0"/>
        <xs:element name="data_type" minOccurs="0">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="number"/>
                    <xs:enumeration value="string"/>
                    <xs:enumeration value="raw"/>
                    <xs:enumeration value="integer number"/>
                    <xs:enumeration value="real number"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="id" type="xs:Name"/>
    <xs:attribute name="ref" type="xs:Name"/>
</xs:complexType>
<xs:complexType name="BehaviorModelsType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="steadyStateResponse"
type="SteadyStateResponseType"/>
        <xs:element name="impulseResponse" type="ImpulseResponseType"/>
        <xs:element name="frequencyResponse" type="FrequencyResponseType"/>
        <xs:element name="latencyTime" type="LatencyTimeBehavioralType"/>
        <xs:element name="temporalOffset" type="TemporalBehavioralType"/>
        <xs:element name="integrationTime" type="TemporalBehavioralType"/>
        <xs:element name="noiseFigure" type="NoiseFigureBehavioralType"/>
        <xs:element name="spatialGeometry"
type="SpatialGeometryBehavioralType"/>
        <xs:element name="other" type="xs:anyType"/>
```



```
        </xs:choice>
        <xs:attribute name="dataUnitUIDRef" type="xs:anyURI"/>
    </xs:complexType>
    <!--ClusterDescription Entry Types-->
    <xs:complexType name="EncodeArrayType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="0"/>
            <xs:element name="arraySize" type="xs:integer"/>
            <xs:element name="indexSequence" type="BindType" minOccurs="0"/>
            <xs:choice>
                <xs:element name="encodeUnit" type="EncodeUnitType"/>
                <xs:element name="encodeSet" type="EncodeSetType"/>
                <xs:element name="encodeArray" type="EncodeArrayType"/>
            </xs:choice>
        </xs:sequence>
        <xs:attribute name="id" type="xs:Name"/>
        <xs:attribute name="ref" type="xs:Name"/>
        <xs:attribute name="encodingType">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="binary_base64"/>
                    <xs:enumeration value="binary_base16"/>
                    <xs:enumeration value="text_delimited"/>
                    <xs:enumeration value="text_fixed_width"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="EncodeSetType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="0"/>
            <xs:element name="indexSequence" type="BindType" minOccurs="0"/>
            <xs:choice maxOccurs="unbounded">
                <xs:element name="encodeUnit" type="EncodeUnitType"/>
                <xs:element name="encodeSet" type="EncodeSetType"/>
                <xs:element name="encodeArray" type="EncodeArrayType"/>
            </xs:choice>
        </xs:sequence>
        <xs:attribute name="id" type="xs:Name"/>
        <xs:attribute name="ref" type="xs:Name"/>
        <xs:attribute name="encodingType">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="binary_base64"/>
                    <xs:enumeration value="binary_base16"/>
                    <xs:enumeration value="text_delimited"/>
                    <xs:enumeration value="text_fixed_width"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="EncodeUnitType">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="0"/>
            <xs:element name="indexSequence" type="BindType" minOccurs="0"/>
            <xs:element name="encodingDetails" type="EncodingDetailsType"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:Name"/>
        <xs:attribute name="ref" type="xs:Name"/>
        <xs:attribute name="encodingType">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="binary_base64"/>
                    <xs:enumeration value="binary_base16"/>
                    <xs:enumeration value="text_delimited"/>
```



```
                    <xs:enumeration value="text_fixed_width"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="EncodingDetailsType">
        <xs:sequence>
            <xs:element name="fieldOffset" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="chars" type="BindType" minOccurs="0"/>
                        <xs:element name="bytes" type="BindType" minOccurs="0"/>
                        <xs:element name="bits" type="BindRefType"
minOccurs="0"/>
                    </xs:sequence>
                    <xs:attribute name="offsetRef" type="xs:Name"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="fieldWidth" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="chars" type="BindType" minOccurs="0"/>
                        <xs:element name="bytes" type="BindType" minOccurs="0"/>
                        <xs:element name="bits" type="BindType" minOccurs="0"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="textFieldType" minOccurs="0">
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="xs:string">
                            <xs:attribute name="beginningDelimiter"
type="xs:string"/>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="binaryFieldType" minOccurs="0">
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="xs:string">
                            <xs:attribute name="endian">
                                <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                        <xs:enumeration value="little"/>
                                        <xs:enumeration value="big"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:attribute>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <!--Behavior Models Types-->
    <xs:complexType name="SteadyStateResponseType">
        <xs:sequence>
            <xs:element name="phenAxis" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="TransferFunctionAxisType">
                            <xs:sequence minOccurs="0">
                                <xs:element name="phenType" type="xs:string"/>
```



```
                            <xs:element name="calibration" type="xs:string"
minOccurs="0"/>
                        </xs:sequence>
                        <xs:attribute name="units"/>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
        </xs:element>
        <xs:element name="dataAxis" maxOccurs="unbounded">
            <xs:complexType>
                <xs:complexContent>
                    <xs:extension base="TransferFunctionAxisType">
                        <xs:sequence minOccurs="0">
                            <xs:element name="calibration"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
        </xs:element>
        <xs:element name="continuity" minOccurs="0">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="discrete"/>
                    <xs:enumeration value="continuous"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="interpolation" minOccurs="0">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="undefined"/>
                    <xs:enumeration value="returnToZero"/>
                    <xs:enumeration value="lastValue"/>
                    <xs:enumeration value="interpolate"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="ImpulseResponseType">
    <xs:choice maxOccurs="unbounded">
        <xs:element name="amplitudeAxis" type="TransferFunctionAxisType"/>
        <xs:element name="timeAxis" type="TransferFunctionAxisType"/>
    </xs:choice>
</xs:complexType>
<xs:complexType name="FrequencyResponseType">
    <xs:choice maxOccurs="unbounded">
        <xs:element name="amplitudeAxis" type="TransferFunctionAxisType"/>
        <xs:element name="frequencyAxis" type="TransferFunctionAxisType"/>
        <xs:element name="phaseAxis" type="TransferFunctionAxisType"/>
        <xs:element name="powerAxis" type="TransferFunctionAxisType"/>
    </xs:choice>
    <xs:attribute name="type">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="carrier"/>
                <xs:enumeration value="modulation"/>
                <xs:enumeration value="powerSpectralDensity"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:complexType>
<xs:complexType name="LatencyTimeBehavioralType">
    <xs:complexContent>
        <xs:extension base="ValueAccuracyType"/>
```



```
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="TemporalBehavioralType">
        <xs:choice>
            <xs:element name="absoluteTime" type="TimeValueAxis" minOccurs="0"/>
            <xs:element name="relativeTime" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="TimeValueAxis">
                            <xs:sequence>
                                <xs:element name="referencePeriodLength"
type="BindType"/>
                                <xs:element name="periodLength" type="BindType"/>
                            </xs:sequence>
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:choice>
    </xs:complexType>
    <xs:complexType name="SpatialGeometryBehavioralType">
        <xs:sequence>
            <xs:element name="identification" type="IdentificationType"
minOccurs="0"/>
            <xs:element name="ambShape" type="AmbiguityShapeType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="ambShapeFramePosition" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="posCoordAxis" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:complexContent>
                                    <xs:extension
base="PositionCoordinateAxisType">
                                        <xs:attribute name="coordName">
                                            <xs:simpleType>
                                                <xs:restriction base="xs:string">
                                                    <xs:enumeration value="x"/>
                                                    <xs:enumeration value="y"/>
                                                    <xs:enumeration value="z"/>
                                                    <xs:enumeration value="alpha"/>
                                                    <xs:enumeration value="beta"/>
                                                    <xs:enumeration value="rho"/>
                                                    <xs:enumeration value="omega"/>
                                                    <xs:enumeration value="phi"/>
                                                    <xs:enumeration value="kappa"/>
                                                </xs:restriction>
                                            </xs:simpleType>
                                        </xs:attribute>
                                    </xs:extension>
                                </xs:complexContent>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                    <xs:attribute name="coordSystem">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:enumeration value="spherical"/>
                                <xs:enumeration value="rectangular"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:attribute>
                    <xs:attribute name="relToRefSys">
                        <xs:simpleType>
```



```
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="transducer"/>
                            <xs:enumeration value="earth"/>
                            <xs:enumeration value="localEarth"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="UIDref" type="xs:Name"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="transAperSize" type="TransducerAperatureSizeType"
minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="ambShapeType">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="ifoi"/>
                <xs:enumeration value="processedShape"/>
                <xs:enumeration value="externalSurface"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:complexType>
<xs:complexType name="AmbiguityShapeType">
    <xs:sequence>
        <xs:element name="name" type="xs:string" minOccurs="0"/>
        <xs:element name="probablity" type="BindType" minOccurs="0"/>
        <xs:element name="PointCloudList" minOccurs="0">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="pointCoordAxis" minOccurs="0"
maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:extension base="ValueArrayAxisType">
                                    <xs:attribute name="coordName">
                                        <xs:simpleType>
                                            <xs:restriction base="xs:string">
                                                <xs:enumeration value="x"/>
                                                <xs:enumeration value="y"/>
                                                <xs:enumeration value="z"/>
                                                <xs:enumeration value="alpha"/>
                                                <xs:enumeration value="beta"/>
                                                <xs:enumeration value="rho"/>
                                            </xs:restriction>
                                        </xs:simpleType>
                                    </xs:attribute>
                                </xs:extension>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="coordSystem">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="spherical"/>
                            <xs:enumeration value="rectangular"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:complexType>
        </xs:element>
        <xs:element name="ambShape" type="AmbiguityShapeType" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
```



```
        <xs:attribute name="shapeType">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="solid"/>
                    <xs:enumeration value="invSolid"/>
                    <xs:enumeration value="surface"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="id" type="xs:Name"/>
        <xs:attribute name="ref" type="xs:Name"/>
    </xs:complexType>
    <xs:complexType name="TransducerAperatureSizeType">
        <xs:sequence>
            <xs:element name="crossSectionArea" type="ValueArrayAxisType"/>
            <xs:element name="pointCoordAxis" minOccurs="0"
maxOccurs="unbounded">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="ValueArrayAxisType">
                            <xs:attribute name="coordName">
                                <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                        <xs:enumeration value="x"/>
                                        <xs:enumeration value="y"/>
                                        <xs:enumeration value="z"/>
                                        <xs:enumeration value="alpha"/>
                                        <xs:enumeration value="beta"/>
                                        <xs:enumeration value="rho"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:attribute>
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="coordSystem">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="spherical"/>
                    <xs:enumeration value="rectangular"/>
                    <xs:enumeration value="wgs84geodetic"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="ValueArrayAxisType">
        <xs:sequence>
            <xs:element name="sequence" type="BindType" minOccurs="0"/>
            <xs:element name="multiplier" type="BindType" minOccurs="0"/>
            <xs:element name="offset" type="BindType" minOccurs="0"/>
            <xs:element name="accuracy" type="AccuracyType" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="PositionCoordinateAxisType">
        <xs:choice>
            <xs:element name="value" type="BindType"/>
            <xs:element name="valueAxis" type="PositionCoordinateAxisType"
maxOccurs="unbounded"/>
        </xs:choice>
        <xs:attribute name="ambShapeRef" type="xs:Name"/>
        <xs:attribute name="index"/>
        <xs:attribute name="axis"/>
```



```
    </xs:complexType>
    <xs:complexType name="NoiseFigureBehavioralType">
        <xs:sequence>
            <xs:element name="value" type="ValueAxisType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="TimeValueAxis">
        <xs:choice>
            <xs:element name="timeValue" type="ValueAccuracyType"/>
            <xs:element name="valueAxis" type="TimeValueAxis"
maxOccurs="unbounded"/>
        </xs:choice>
        <xs:attribute name="axis"/>
        <xs:attribute name="index"/>
    </xs:complexType>
    <!--Relation Entry Types-->
    <xs:complexType name="AnnotationRelationType">
        <xs:sequence>
            <xs:element name="documentation" minOccurs="0">
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="xs:string">
                            <xs:attribute name="source"/>
                            <xs:attribute name="UIDref" type="xs:Name"/>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="UIDref" type="xs:Name"/>
    </xs:complexType>
    <xs:complexType name="AttachRelationType">
        <xs:sequence>
            <xs:element name="attachDesc" type="xs:string" minOccurs="0"/>
            <xs:element name="value" type="BindRefType"/>
        </xs:sequence>
        <xs:attribute name="bindUIDref" type="xs:Name"/>
    </xs:complexType>
    <xs:complexType name="PositionRelationType">
        <xs:sequence>
            <xs:element name="posCoord" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="ValueAccuracyType">
                            <xs:attribute name="coordName" use="required">
                                <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                        <xs:enumeration value="x"/>
                                        <xs:enumeration value="y"/>
                                        <xs:enumeration value="z"/>
                                        <xs:enumeration value="alpha"/>
                                        <xs:enumeration value="beta"/>
                                        <xs:enumeration value="rho"/>
                                        <xs:enumeration value="omega"/>
                                        <xs:enumeration value="phi"/>
                                        <xs:enumeration value="kappa"/>
                                        <xs:enumeration value="lat"/>
                                        <xs:enumeration value="long"/>
                                        <xs:enumeration value="alt"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:attribute>
                            <xs:attribute name="uom">
                                <xs:simpleType>
                                    <xs:restriction base="xs:string"/>
```



```
                    </xs:simpleType>
                </xs:attribute>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="velCoord" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="ValueAccuracyType">
                <xs:attribute name="coordName" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="x"/>
                            <xs:enumeration value="y"/>
                            <xs:enumeration value="z"/>
                            <xs:enumeration value="alpha"/>
                            <xs:enumeration value="beta"/>
                            <xs:enumeration value="rho"/>
                            <xs:enumeration value="omega"/>
                            <xs:enumeration value="phi"/>
                            <xs:enumeration value="kappa"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="uom">
                    <xs:simpleType>
                        <xs:restriction base="xs:string"/>
                    </xs:simpleType>
                </xs:attribute>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="accelCoord" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="ValueAccuracyType">
                <xs:attribute name="coordName" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="x"/>
                            <xs:enumeration value="y"/>
                            <xs:enumeration value="z"/>
                            <xs:enumeration value="alpha"/>
                            <xs:enumeration value="beta"/>
                            <xs:enumeration value="rho"/>
                            <xs:enumeration value="omega"/>
                            <xs:enumeration value="phi"/>
                            <xs:enumeration value="kappa"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="uom">
                    <xs:simpleType>
                        <xs:restriction base="xs:string"/>
                    </xs:simpleType>
                </xs:attribute>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="transducerUIDref" type="xs:Name" use="required"/>
<xs:attribute name="coordSystem">
```



```
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="spherical"/>
                    <xs:enumeration value="rectangular"/>
                    <xs:enumeration value="wgs84geodetic"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="relativeRefSys">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="earth"/>
                    <xs:enumeration value="local"/>
                    <xs:enumeration value="transducer"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="relToCompUIDref" type="xs:Name"/>
    </xs:complexType>
    <xs:complexType name="TimeDatumType">
        <xs:choice minOccurs="0">
            <xs:element name="timestamp" type="BindRefType"/>
            <xs:sequence>
                <xs:element name="year" type="BindRefType"/>
                <xs:element name="month" type="BindRefType"/>
                <xs:element name="day" type="BindRefType"/>
                <xs:element name="hour" type="BindRefType"/>
                <xs:element name="min" type="BindRefType"/>
                <xs:element name="sec" type="BindRefType"/>
            </xs:sequence>
        </xs:choice>
        <xs:attribute name="sysClkRef" type="xs:Name"/>
        <xs:attribute name="timeDatum">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="UTC"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="ref" type="xs:Name"/>
    </xs:complexType>
    <xs:complexType name="ConnectNodeRelationType">
        <xs:sequence>
            <xs:element name="connectDescription" type="xs:string"/>
            <xs:element name="connectUID" type="BindRefType"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="FeatureOfInterestRelationType">
        <xs:sequence>
            <xs:element name="identification" minOccurs="0"
maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="description" type="xs:string"/>
                    </xs:sequence>
                    <xs:attribute name="UIDref" type="xs:Name"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="UIDRef" type="xs:Name"/>
    </xs:complexType>
    <xs:complexType name="RemoteEnergyRelationType">
        <xs:sequence>
            <xs:element name="medium">
```



```
            <xs:complexType>
                <xs:simpleContent>
                    <xs:extension base="xs:string">
                        <xs:attribute name="propagationMechanism">
                            <xs:simpleType>
                                <xs:restriction base="xs:string">
                                    <xs:enumeration value="radiation"/>
                                    <xs:enumeration value="conduction"/>
                                    <xs:enumeration value="convection"/>
                                    <xs:enumeration value="osmosis"/>
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:attribute>
                    </xs:extension>
                </xs:simpleContent>
            </xs:complexType>
        </xs:element>
        <xs:element name="reflEnergySource">
            <xs:complexType>
                <xs:simpleContent>
                    <xs:extension base="xs:string">
                        <xs:attribute name="energySourceType">
                            <xs:simpleType>
                                <xs:restriction base="xs:string">
                                    <xs:enumeration value="natural"/>
                                    <xs:enumeration value="artificial"/>
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:attribute>
                        <xs:attribute name="reflPhenSource" type="xs:anyURI"/>
                    </xs:extension>
                </xs:simpleContent>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="UIDRef" type="xs:Name"/>
    <xs:attribute name="type">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="reflected"/>
                <xs:enumeration value="emitted"/>
                <xs:enumeration value="both"/>
                <xs:enumeration value="unknown"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:schema>
```


Bibliography

[1] Manual of Photogrammetry, American Society of Photogrammetry, Fifth Edition, 2004

[2] Optical & Photographic Reconnaissance Systems, Niels Jensen, 1968

[3] Handbook of Modern Sensors – Physics, Designs, and Applications (3rd Edition), Jacob Fraden, 2003

[4] Handbook of Measurement Science, Volume 1 Theoretical Fundamentals, Peter Sydenham, 1992

[5] Handbook of Measurement Science, Volume 2 Practical Fundamentals, Peter Sydenham, 1992

[6] Handbook of Measurement Science, Volume 3 Elements of Change, Peter Sydenham, 1992

[7] Acoustical Imaging Volume 19, Proceedings of the 19th International Symposium on Acoustical Holography, 1991

[8] Fundamentals of Electronic Instrumentation for Measurement, First Edition, William Ribbins, 1973

[9] Introduction to Airborne Radar, George Stimson, 1983

[10] Multisensor Data Fusion, Ed Waltz, James Llinas, 1990

[11] Mathematical Techniques in Multisensor Data Fusion, David Hall, 1992

[12] Elements of Photogrammerty, Paul Wolf, 1974

[13] Introduction to Radar Systems, Merrill Skolink, 1980

[14] Synthetic Aperture Radar Signal Processing, Mehrdad Soumekh, 1999

[15] RADAR Handbook (Second Edition), Merrill Skolnik, 1990

[16] Radargrammetric Image Processing, Franz Leberl, 1989

[17] Communication and Radar Systems, Nicolaos Tzanes, 1985

[18] Synthetic Aperture Radar – Systems and Signal Processing, John Curlander, Robert McDonough, 1991

[19] Signals, Systems and Transforms, Charles Phillips, John Parr, 1995

[20] Introduction to Strapdown International Navigation Systems, Paul Savage, 1985

[21] DMA TR 8350.2-B Supplement to DOD WGS-84 Technical Report – Parameters, Formulas, and Graphics for the Practical Application of WGS-84, 1987

[22] Proposed Replacement Sensor Model (RSM) NITF TRE, Draft 7/23/2004

