.

Kizoom Limited, 109-123 Clifton Street, London EC2A 4LD. Tel: +44 207 749 2670

UTMC XML Handbook

Real-time exchange services for UTMC data over SIRI & Bulk Publication services

Version 0.08

2009/07/22 Njsk Kizoom

REVISED DRAFT

UTMX XML -Handbook

Control sheet

Version control

Date	Author	Version	Description of changes
2009/03/20	Nick Knowles,	V.01	Created from SIRI handbook
2009/04/15	Nick Knowles,	V.02	Internal Revised with diagrams
2009/06/02	Nick Knowles,	V.03	Revised from XML
2009/06/05	Nick Knowles,	V.04	Coverage of Publication
2009/06/05	Nick Knowles,	V.06	Coverage of Publication
2009/07/22	Nick Knowles,	v.08	Add cctv ANPRVrnDynamic

Document automation & Copyright notice

Kizoom Ltd
107-109 Clifton Street,
London EC2A 4LD

Telephone: 207 749 2670

Company Registration Number : 3745127

© Copyright 2007, 2009 Freeflow , Kizoom :

All rights Reserved.

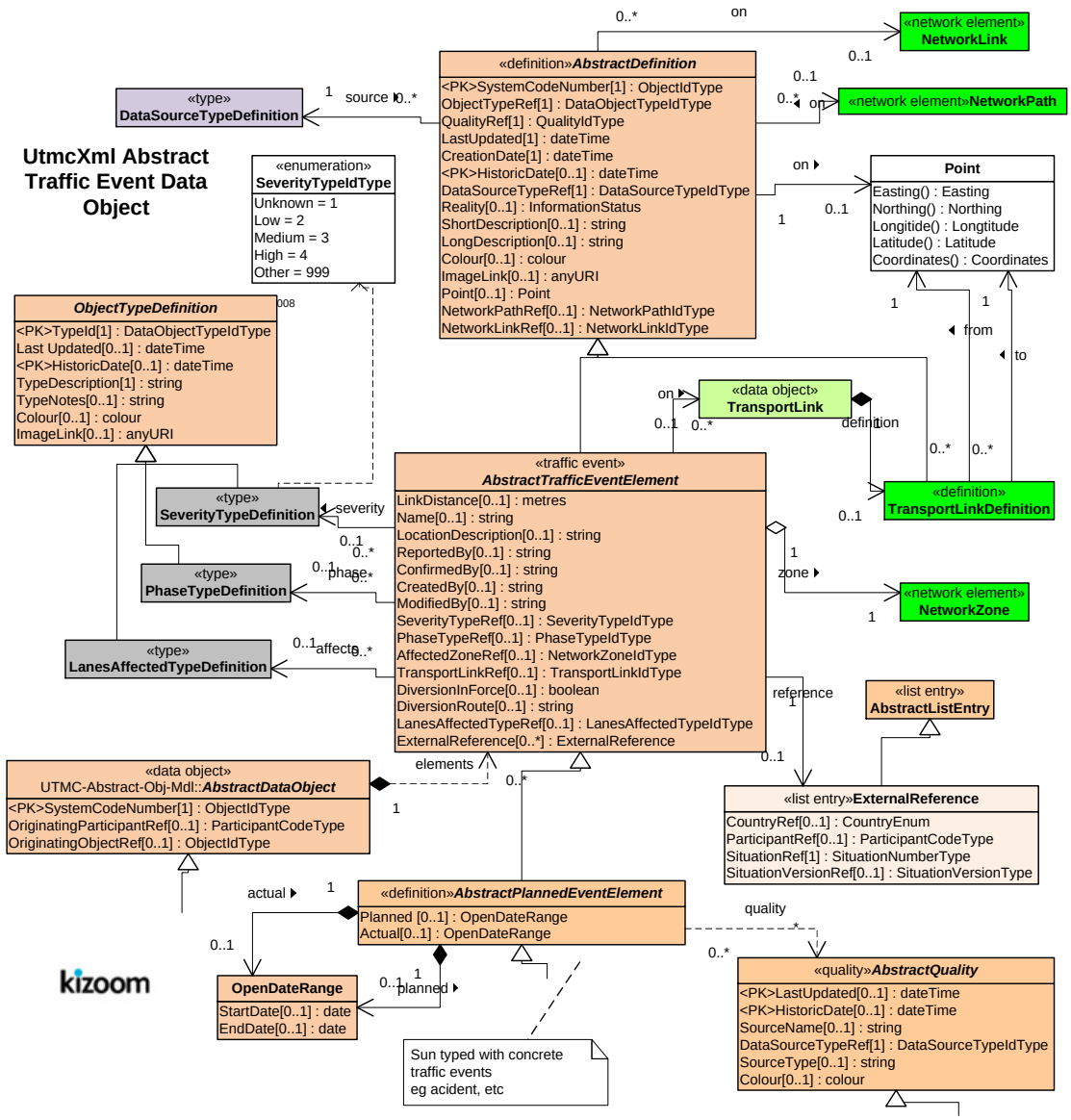
Author:

Nicholas Knowles

Table of Contents

© Copyright Kizoom 2007-2009

© Copyright Kizoom 2007-2009



© Copyright Kizoom 2007-2009

© Copyright Kizoom 2007-2009

© Copyright Kizoom 2007-2009

1. INTRODUCTION

This UTMC XML Handbook provides a short overview of the UTMC XML protocol (**UtmcXml**) which provides an XML Interface for exchanging UTMC data between servers. UtmcXml can be used both for the fine grained real-time exchange of objects, using the common framework **for real-time data exchange** provided by the CEN SIRI (Service Interface for Real Time Information) technical specification, and for the Bulk exchange of UTMC data using the same XML objects in a **Publication** protocol.

The Handbook is intended to provide both an introduction to UTMC-XML for technical readers, and a quick reference guide for developers who are already familiar with the service, but wish to check particular points of detail. It does not cover a number of more detailed aspects of UTMC or SIRI (for example capability discovery, access controls, etc).

A particular purpose of the handbook is to illustrate the relationship between the UtmcXml and the UTMC UML model by diagramming the UtmcXml message content in such a way that the reference to the UML model is explicit.

Another goal of this handbook is make understanding a simple use of UtmcXml for common applications as accessible as possible. As technical standards intended to support many different implementations and many different applications and usages, both UTMC and SIRI have a lot of features and optional capabilities. Consequently they require relatively large specifications. Nonetheless, commonly required basic real-time services, such as traffic flows or network link can be understood and implemented simply in UtmcXml, and this guide is intended to highlight such straightforward uses.

The guide assumes a technical knowledge of XML and web services. and of data modelling with UML notation, as well as a reasonable knowledge if UTMC..

ACKNOWLEDGEMENTS

Work on the UTMC XML schema has been supported by a Technology Strategy Board grant.

The UDG technical committee has provided vital technical input into the UTMC XML development and the XML representational model has drawn heavily of the work by on the UTMC UML model , which was created by Alastair Dunsmore, Ian Cornwell and Craig Palmer of Mott MacDonald, with review inputs from Jim David, (Alison Jones (Siemens)), Brendan Mason (Envitia) and Will Pickering (Amey).

The Freeflow Objects have been developed by the Freeflow consortium with input form ....

This supporting document is derived on the SIRI Handbook which was developed by Kizoom from the SIRI specification. The SIRI specification was developed with sponsorship from VDV, RTIG, Department of Transport, CERTU and other organizations.

REFERENCES

- UTMC
    - UTMC Data Object specification: ts004.003annexdapr07.xls

© Copyright Kizoom 2007-2009

  - UTMC Data Object specification: ts004.003annexdapr07-XML-09.xls: the above annotated with XML mappings
  - UTMC UML specification: ts004-004a-annexd-sep08.pdf
  - UTMC XML SIRI schema – utm,c.xsd
  - UTMC XML SIRI schema – utmc_publication.xsd
- SIRI Specification Service interface for real-time information relating to public transport operations
  - CEN/TS 00278181-1 — Part 1: Introduction
  - CEN/TS 00278181-2 — Part 2: Communications infrastructure
  - CEN/TS 00278181-3 — Part 3: Functional Service Interfaces
- SIRI Schema
  - http://www.siri.org.uk/schema/1.2/siri.xsd
- SIRI White Paper
  - http://www.siri.org.uk/
- Transmodel
  - CEN TC 278 ENV 12896 Reference Data Model for Public Transport

© Copyright Kizoom 2007-2009

2. INTRODUCTION TO THE UTMC XML API

WHAT ARE UTMC DATA OBJECTS?

The Urban Traffic Management & Control (UTMC) common database provides a set of "UTMC Data Objects" appropriate for representing many aspects of road traffic management, including traffic flows on road links, VMS sign content, road works, accidents, incidents, measurement device data, CCTV camera content, car park data, etc . These objects fit into a common architecture that allows for static, configurable and dynamic contenta aspects of each object type.

Table 2-1 shows the Data Objects included in UtmcXml. The objects fall into several distinct groups

(i) **Transport Link** & Route objects - providing flows on links & routes. Links may have both a static and a dynmic part

(ii) **Device** Objects - providing different types of data for objects located at particular points, such as detectors, VMS signs, car parks; Devices may accept commands to operate them – for example to swivel a camera, and have components for monitoring and fault management. They may be located on the network by references to through a **Network Path** or **NetworkLink**

(iii) **Traffic Event** Objects - describing different types of disruption event, such as accidents and roadworks, and;

(iv) **Network** definition objects, describing the fixed network and its geometry, such as nodes, links and turns. For certain objects prediction and historic profile objects may be associated. These objects seldom change and provide a reference context for the other objects.

(v) **Prediction & Profile** objects, providing summaries of expected values of dynamic properties of the network such as speeds or flows a given time; these are either based on historic data or the outputs of various predictive methods.

In addition to the functional objects given above there are a number of ancillary components to specify quality parameters, status values, equipment types etc.

In general, the UTMC objects follow a very regular structural pattern; each Data Object has common base attributes and instantiates the UTMC framework with a similar set of component parts (e.g. for a **Device**, a **Definition**, **Configuration**, and **Dynamic** parts). Some object types also extend the framework with elements specific to the type of Data Object (Shown in dark blue font)

© Copyright Kizoom 2007-2009

Group	Data Object	Componenets	Profile & Prediction
Link	**Transport Link**	Definition, *Configuration* (ANPR, GPS SCOOT), *Dynamic* (ANPR, GPS SCOOT), History, Fault, Command	Profile. Prediction
	Transport Route	Definition, *Configuration* (Route, Route Segment), Dynamic, History, Fault, Command	Profile. Prediction
Device	**Access Control**	Definition, Configuration, Dynamic, History, Fault, Command	Profile. Prediction
	Air Quality	Definition, Configuration, Dynamic, History, Fault, Command	Profile. Prediction
	Car Park	Definition, *Configuration* (Capacity, Detector, Opening Times, Tariff), Dynamic, History, Fault, Command Car Park Access Location	Profile. Prediction
	Close Circuit TV	Definition, Configuration, *Dynamic* (ANPR Camera, Broadcast, Flow, Image, Queue), History, Fault, Command	Profile. Prediction
	Detector	Definition, Configuration, *Dynamic* (Dynamic Flow, Headway, Occupancy, Queue, Speed), History, Fault, Command	Profile. Prediction
	Meteorological	Definition, Configuration, Dynamic, History, Fault, Command	--
	Traffic Signal	Definition, Configuration, Dynamic, History, Fault, Command	Profile. Prediction
	Variable message Sign	Definition, Configuration, Dynamic, History, Fault, Command, Message List, Car Park List	--
Traffic Event	**Accident**	Element, Accident Statistics	--
	Incident	Element	--
	Planned Event	Element	--
	Roadworks	Element	--
Network	**Network Geometry**	Object, Points	--
	Network Link	Object, Nodes, Geometry, Paths, Zones	--
	Network Node	Object, Geometry	--
	Network Path	Object, List of Links	--
	Network Turn	Object, Links	--
	Network Zone	Object, List of Links	--

Table 2-1 UTMC Data Objects

© Copyright Kizoom 2007-2009

Each Data Object may reference a number of Type Definition Objects (Table 2-2) that provide representations of different allowed properties of the Data Object, including an object subtype a .

Group	Data Object	Data Object Type	Generic Types	Data Object Specific Types
Link	**Transport Link**	Transport Link Type	Data Source, Subsystem Type, Command Format Type	Quality, Fault Type, Transport Link Status Type
	Transport Route	Transport Route Type		Quality, Fault Type, Transport Route Status Type
Device	**Access Control**	Access Control Type	Data Source, Subsystem Type, Command Format Type	Quality, Fault Type, Access Control State Type
	Air Quality	Air Quality Type		Quality, Fault Type
	Car Park	Car Park Type		Quality, Fault Type, Car Park State Type Car Park Trend Type
	Close Circuit TV	CcTV Type		Quality, Type, CCTV Vehicle Type
	Detector	Detector Type		Quality, Fault Type, Flow Status Type, Headway Status Type, Occupancy Status Type, Queue Severity Type, Speed Status Type
	Meteorological	Meteorological Type		Quality, Fault Type, Precipitation Type, Road Condition Type, Visibility Type
	Traffic Signal	Traffic Signal Type		Quality, Fault Type
	Variable message Sign	Variable message Sign Type		Quality, Fault Type
Traffic Event	**Accident**	Accident Type	Data Source, Lanes Affected Type Severity Type, Phase Type	Quality
	Incident	Incident Type		Quality
	Planned Event	Event Type		Quality
	Roadworks	Roadworks Type		Quality
Network	**Network Geometry**	--	--	--
	Network Link	--	--	--
	Network Node	--	--	--
	Network Path	Network Path Type	--	--
	Network Turn	--	--	--
	Network Zone	Network Zone Type	--	--

Table 2-2 UTMC Data Object related Types

© Copyright Kizoom 2007-2009

WHAT IS THE UTMCXML XML API?

Traditionally a CORBA interface has been used to exchange data between UTMC databases, requiring the use of CORBA ORB technology and IDL. While CORBA is an efficient and appropriate technology for many usages, it requires a fairly close degree of coupling between systems, making it harder to integrate diverse distributed systems. Neither does it readily support some of the filtering requirements found in common web service use.

The UtmcXml API provides a means of exchanging the same UTMC data using an XML representation. This allows the use of mainstream XML technology for processing and exchanging data and can be used in both synchronous and asynchronous protocols. It also adds additional semantics fro querying the UTMC database intended to reflect common web service use.

For each UTMC Data Object, UtmcXml specifies an XML rendering, that is a serialisation of the object as a set of XML tags and values. XML data typing and validation mechanisms can be used to specify a number of cross checks that improve data quality and reduce the chances of mistakes. The XML representation can be reused in many different ways

To exchange the XML objects two different protocols are supported

(i) **UtmcXml-SIRI**. [NAME?] or SIRI-UTMC-DO An exchange format based on the CEN SIRI Interface framework. This adds a SIRI conformant web service that allows for both request/responses and publish/subscribe (i.e. event driven) exchange of one or many UTMC data objects, using standard http or SOAP mechanisms.

(ii) **UtmcXml-Publication**. NAME?]A bulk exchange format that allows the exchange of UTMC objects as of objects.

The XML API is pretested against a number of the main XML tools

WHY USE XML?

XML allows for the self-describing representation that can be used flexibly in many different ways. Benefits include:

- It supports a fairly high level of abstraction that captures many constraints for validation and referential integrity as well as rich documentation.
- It is possible to use the same interface for both full and partial implementations, allowing an incremental uptake. An extended set of UTMC elements has been included in the interface, but an implementation may choose to omit elements that it does need to support
- Schemas can include fine grained versioning and the ability to support different versions concurrently (allowing staggered upgrades)
- It is a mainstream technology with widespread tool support on all platforms, open source tools, and readily available skills

© Copyright Kizoom 2007-2009

IMPLEMENTING UTMC XML – SOME INTRODUCTORY EXAMPLES

The UtmcXml XML is intended to be straightforward to use with current actual mainstream XML tools and technology that automate the binding process. Slightly different considerations apply to (a) outputting data to an XML document and (b) reading data from an XML document.

USING UTMCXML FOR OUTPUT FROM AN APPLICATION

To output UtmcXml, an implementer will typically write a program to iterate over their UTMC data base and output an XML stream containing the required tags and values for the requested Data Objects. UtmcXml uses consistent design pattern for serialising UTMC objects so a generic application can be used, with a simple adaptor for each Data Object to map its specific data elements. This can also include the object specific filtering logic to select the desired content

For example here are fragment of parts of two different Data Objects, a Transport Link and a Car Park

EXAMPLE FRAGMENT OF A TRANSPORT LINK

```
<TransportLink ">
        <SystemCodeNumber>TL1234</SystemCodeNumber>
        <TransportLinkDefinition>
                <ObjectTypeRef>SCOOT</ObjectTypeRef>
                <QualityRef>Qual01</QualityRef>
                <CreationDate>2001-12-17T09:30:47.0Z</CreationDate>
                <DataSourceTypeRef>QMISS</DataSourceTypeRef>
                <ShortDescription>Park Lane Northbound</ShortDescription>
                <Point>
                        <Easting>999999</Easting> <Northing>1999999</Northing>
                </Point>
                <EndPoint >
                        <Easting>999999</Easting><Northing>1999999</Northing>
                </EndPoint>
                <NetworkPathRef>Net001</NetworkPathRef>
        </TransportLinkDefinition>

        <Dynamics>
                <TransportLinkGpsDynamic>
                        <LastUpdated>2001-12-17T09:30:47.0Z</LastUpdated>
                        <AverageSpeed>30.5</AverageSpeed>
                        <LinkTravelTime>PT3M0S</LinkTravelTime>
                        <LinkStatusTypeRef>normal</LinkStatusTypeRef>
                        <CurrentJourneyTime>PT3M0S</CurrentJourneyTime>
                        <MeanTravelTime>PT3M30.5S</MeanTravelTime>
                        <MedianTravelTime>PT3M0S</MedianTravelTime>
                        <NumberOfVehicles>1000</NumberOfVehicles>
                </TransportLinkGpsDynamic>
                <TransportLinkGpsDynamic>
                        <LastUpdated>2001-12-17T09:30:47.0Z</LastUpdated>
                        <AverageSpeed>30.5</AverageSpeed>
                        <LinkTravelTime>PT3M0S</LinkTravelTime>
                        <LinkStatusTypeID>normal</LinkStatusTypeID>
                        <CurrentJourneyTime>PT3M0S</CurrentJourneyTime>
                        <MeanTravelTime>PT3M30.5S</MeanTravelTime>
                        <MedianTravelTime>PT3M0S</MedianTravelTime>
                        <VarianceTravelTime>PT30S</VarianceTravelTime>
                        <NumberOfVehicles>1000</NumberOfVehicles>
                        <ReservedLane>20</ReservedLane>
                </TransportLinkGpsDynamic>
        </Dynamics>
        <DeviceHistory>
                <TransportLinkDeviceHistory>
.....
                </TransportLinkDeviceHistory>
        </DeviceHistory>
        <Faults>
                <TransportLinkFault>
```

© Copyright Kizoom 2007-2009

```
....
                    </TransportLinkFault>
          </Faults>
          <Commands>
                    <TransportLinkCommand>
                              <RequestTime>2001-12-17T09:30:47.0Z</RequestTime>
.....
                    </TransportLinkCommand>
          </Commands>
</TransportLink>
```

EXAMPLE FRAGMENT OF A CAR PARK\

```
<?xml version="1.0" encoding="UTF-8"?>
<CarPark version="1.0" xsi:schemaLocation="http://www.utmc.uk.com/utmc ../../utmc_publication.xsd"
xmlns="http://www.utmc.uk.com/utmc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
          <ParticipantRef>1234</ParticipantRef>
          <SystemCodeNumber>CP001</SystemCodeNumber>
          <CarParkDefinition>
                    <ObjectTypeRef>String</ObjectTypeRef>
                    <QualityRef>String</QualityRef>
                    <CreationDate>2001-12-17T09:30:47.0Z</CreationDate>
                    <DataSourceTypeRef>String</DataSourceTypeRef>
                    <ShortDescription>a</ShortDescription>
                    <LongDescription>a</LongDescription>
                    <Point
                              <Easting>999999</Easting>
                              <Northing>1999999</Northing>
                    </Point>
                    <NetworkPathRef>NP1234</NetworkPathRef>
                    <TransportLinkRef>tl25698</TransportLinkRef>
                    <LinkDistance>0.0</LinkDistance>
                    <OwnerOperator>Foo bar</OwnerOperator>
                    <CarParkUserType>String</CarParkUserType>
                    <NumberOfEntrances>2</NumberOfEntrances>
                    <NumberOfExits>3</NumberOfExits>
                    <ToiletsAvailable>true</ToiletsAvailable>
                    <DisableDataToiletsAvailable>true</DisableDataToiletsAvailable>
                    <AccessLocations>
                              <CarParkAccessLocation>
                                        <EntryOrExit>E</EntryOrExit>
                                        <EntranceLocation id="NMTOKEN" srsName="String">
                                                  <GridType>UKOS</GridType>
                                                  <Easting>999999</Easting>
                                                  <Northing>1999999</Northing>
                                                  </EntranceLocation>
                                        <NetworkPathRef>a</NetworkPathRef>
                                        <TransportLinkRef>a</TransportLinkRef>
                                        <Extensions/>
                              </CarParkAccessLocation>
                    </AccessLocations>
                    <Zones><CarParkZoneListEntry>
                              <NetworkZoneRef>Xon123</NetworkZoneRef>
                    </CarParkZoneListEntry>
                    </Zones>
          </CarParkDefinition>
          <Configurations>
                    <CarParkCapacityConfiguration>
                              <ConfigurationDate>2001-12-17T09:30:47.0Z</ConfigurationDate>
                              <CarParkCapacity>200</CarParkCapacity>
                              <DisabledCapacity>10</DisabledCapacity>
                              <AlmostFullIncreasing>180</AlmostFullIncreasing>
                              <AlmostFullDecreasing>185</AlmostFullDecreasing>
                              <FullIncreasing>190</FullIncreasing>
                              <FullDecreasing>195</FullDecreasing>
                              <EntranceFull>198</EntranceFull>
                              <Extensions/>
                    </CarParkCapacityConfiguration>
                    <CarParkDetectorConfiguration>
                              <ConfigurationDate>2001-12-17T09:30:47.0Z</ConfigurationDate>
                              <DetectorRef>Dt1234</DetectorRef>
                    </CarParkDetectorConfiguration>
                    <CarParkOpeningTimesConfiguration>
```

© Copyright Kizoom 2007-2009

```
                    <ConfigurationDate>2001-12-17T09:30:47.0Z</ConfigurationDate>
                    <Reality>real</Reality>
                    <DayTypeRef>1</DayTypeRef>
                    <OpeningTime>14:20:00.0Z</OpeningTime>
                    <ClosingTime>14:20:00.0Z</ClosingTime>
            </CarParkOpeningTimesConfiguration>
            <CarParkTariffConfiguration>
                    <ConfigurationDate>2001-12-17T09:30:47.0Z</ConfigurationDate>
                    <DayTypeRef>2</DayTypeRef>
                    <TimePeriod>String</TimePeriod>
                    <Cost>50.0</Cost>
            </CarParkTariffConfiguration>
                            <CarParkTariffConfiguration>
                    <ConfigurationDate>2001-12-17T09:30:47.0Z</ConfigurationDate>
                    <DayTypeRef>4</DayTypeRef>
                    <TimePeriod>String</TimePeriod>
                    <Cost>60.0</Cost>
            </CarParkTariffConfiguration>
    </Configurations><Dynamics>
            <CarParkDynamic>
                    <LastUpdated>2001-12-17T09:30:47.0Z</LastUpdated>
                    <CarParkStateTypeRef>String</CarParkStateTypeRef>
                    <Occupancy>0</Occupancy>
                    <OccupancyPercentage>3.14159E0</OccupancyPercentage>
                    <FillRate>0.0</FillRate>
                    <ExitRate>0.0</ExitRate>
                    <QueueTime>P1Y2M3DT10H30M0S</QueueTime>
                    <OccupancyTrendTypeRef>String</OccupancyTrendTypeRef>
                    </CarParkDynamic>
    </Dynamics>

</CarPark>
```

EXCHANGE OF INDIVIDUAL COMPONENTS

In the examples above, the various components of a Data Object are grouped within a "wrapper tag", for the Data Object type (***TransportLink*** or ***CarPark*** respectively) which supplies the SystemCodeNumber common to all components . It is also possible to exchange individual components independently (as with the current CORBA interface) by including the System code number on the individual component.

```
    <TransportLinkGpsDynamic>
            <SystemCodeNumber>TL1234</SystemCodeNumber
            <LastUpdated>2001-12-17T09:30:47.0Z</LastUpdated>
            <AverageSpeed>30.5</AverageSpeed>
            <LinkTravelTime>PT3M0S</LinkTravelTime>
            <LinkStatusTypeRef>normal</LinkStatusTypeRef>
            <CurrentJourneyTime>PT3M0S</CurrentJourneyTime>
            <MeanTravelTime>PT3M30.5S</MeanTravelTime>
            <MedianTravelTime>PT3M0S</MedianTravelTime>
            <NumberOfVehicles>1000</NumberOfVehicles>
    </TransportLinkGpsDynamic>
```

USING UTMCXML FOR INPUT TO AN APPLICATION

Input data from an document in a serialised format (such such as XML) requires parsing it to find the elements of a complex data structure and referencing the links. This is a relatively complex process (whatever the format used) and XML provides tools and technologies that greatly simplify the work an implementor has to do. Typically a developer will use an automatic tool to **bind** to a published schema or subschema, and then write "glue code" to map the parsed objects to his own application objects. The parser will then use the binding to process and validate documents, transforming the validated stream of characters from an Xml document into a correct in-memory representation. The parsere does s the hard work of checking and extracting the data from an XML model. It presents data

© Copyright Kizoom 2007-2009

to the application as a set of data structures that correspond very closely to the existing UTMC data elements.

The UtmcXml has been tested against a number of bindings, e.g. JAVA JAXB

What is the relationship between the UTMC IDL, the UTMC UML & xml?

The UTMC model has a UML representation that provides a conceptual model of the UTMC Data Objects and their relationships – [see UTMC-UML1]. The UML Logical model has been abstracted recently from the original UTMC Object definition of flat table objects [see UTMC-Spc1] and the IDL interface. The UML is conceptual and introduces additional abstractions and clarifications that are not explicit in the flat tables

Both the IDL and the UtmcXml XML representation can be seen as concrete implementations of the abstract representation of the UTMC Logical model.

The UTMC XML schema uses a slightly extended version of the conceptual model that is intended to be functionally equivalent, that is to support all existing UTMC elements and relationships. The XML model is also documented by an UML "Physical" model, one that is very similar to the Definitive UTMC Logical model, but that which also explicitly describes some implementation aspects. . There are thus extensions in the representation for several reasons (i) the syntactic requirements of XML, (ii) minor functional extensions to support Freeflow; (iii) Clarifications of points that are ambiguous in the Logical model – clarifying these improves future proofing . These are discussed below.

Overall the UtmcXml XML schema is intended to provide a stable long term interface for implementers to use. It aims to be consistent and to follow best practice. .

Technical Differences

- Use of more specific simple data types in some cases
- Mapping of some names to fit XML practice or improve consistency,
- Ordering of elements dictated by use of XML reuse of supertypes or groups.
- Modelling of Transport Links & Transport Routes as Devices (ie with Commands, Faults etc) since they may have associated traffic monitoring equipment.

Model differences to enable extensibility

- Introduction of an optional "Wrapper object", with Configuration, Dynamic and Device components as children of this wrapper object rather than of a Definition Object.
- Introduction of a few additional abstract classes to share certain common properties: these are collapsed and repeated in the Standard UTMC UML model.
- Addition of some optional additional attributes – See UtmcXml extensions

UTMC XML STYLE & NAMING CONVENTIONS

© Copyright Kizoom 2007-2009

As with most programming languages, it is possibly in XML to adopt significantly different coding styles and conventions even within the same language. UtmcXml follows a consistent set of guidelines based on established best practice. For example separate Complex structures are used for most XML elements (i.e. "Russian doll" coding is XML eschewed). This gives a higher level of reuse (since individual entities can be reused), and improves clarity (since the correspondence with UML classes is clearer). Elements are generally used only to restrict types.

UTMCXML NAMING CONVENTIONS

1. XML Data Type names
    a. The names of All complex Structures end in "Structure", e.g. ***AccessControlStructure***
    b. Most simple XML types end in Type e.g. ***AccessControlIdType*** (Type is avoided as an element name where possible
    c. Use related names for related subtypes e.g. ***ObjectIdType AccessControlIdType***
    d. Simple fundamental types that are in lower Camel case e.g. speed, distance, boolean, phoneNumber
    e. We avoid Cryptic abbreviations e.g. ***TransportLinkConfiguration*** not ***TR_Configuration.*** This is
2. XML Element Names:
    a. We use Upper Camel Case names for elements, remove "_" e.g. ***AirQuality*** not ***Air_Quality.***
    b. We mostly avoid the use of xxxType or xxxStructure or xxxGroup in element names to avoid confusion with simple types.
    c. Element names typically correspond to that of the structure they use. E.g. a Transport***Link TransportLinkStructure***
    d. We use the suffix xxRef to indicate that an elements represents a reference to the identifier of an element that is being dereferenced in the XML, for example ***TransportLinkRef***. This will have the same type as the identifier of the referenced element, e.g. *TransportLinkIdType*.
    e. We use wrapper tags to encapsulate multiple instances e.g. "Definitions", "Configurations". The name will correspond to that of the model relationship it represents.
    f. Names are generally polymorphic, e.g. Configurations (Not AccessControlConfigurations, DetectorConfigurations, etc)
3. XML Group Names
    a. We use xxxGroup on all groups.
    b. If the group's purpose is to gather the specific elements of a Complex Structure (in order to distinguish from those of the supertype or subtype) , It is named after the structure e.g. TransportLinkStructure/ TransportLinkGroup Where the Group

© Copyright Kizoom 2007-2009

gathers e.g. TransportLinkGroup. Groups are purely syntactic and do not appear in XML documents.

The name mappings from UTMC specification to the XML are given in the Spreadsheet that accompanies the schema. They includes

Abstract Classes (abstract root classes are named as such)

Object_Definition ➔ AbstractDefinition

Object_Configuration ➔AbstractConfiguration

ObjectDynamic ➔AbstractDyunamic

Quality ➔AbstractQuality

Command ➔AbstractCommand

Device_History ➔AbstractDeviceHistory

Fault ➔AbstractFault

Fault_Type ➔AbstractFaultType

GenericElements

Xxx_Commands ➔ XxxCommand ***(***entity should be singular***)***

Xxx_Configuration ➔XxxConfiguration

Xxx_Definition ➔ XxxDefinition

Xxx_Device_History ➔XxxDeviceHistory

Xxx_Faults ➔XxxFault (entity should be singular***)***

Xxx_FaultType ➔XxxFaultType

Xxx_Quality ➔ XxxQuality

Xxx_TypeID ➔ xxxTypeDefinition (Definition is clearer tha ID)

Specific Elements

TR_ ➔ TransportRoute

TL_=> TransportLink

Type_Id ➔TypeDefinition

Headway_Dynamic ➔ DetectorHeadwayDynamic

Occupancy_Dynamic ➔ DetectorOccupancyDynamic

Queue_Dynamic ➔ DetectorQueueDynamic

Speed_Dynamic ➔ SpeedQueueDynamic

© Copyright Kizoom 2007-2009

Car_Par_Tariffs ➔CarParkTariff (entity should be singular***)***

Common Attributes

QualityStatementID➔QualityRef

TypeID ➔ ObjectTypeRef

Xxx_ID ➔ XxxTypeRef

DataSource_TypeID ➔ *DataSourceTypeRef*

NetworkPathReference ➔ NetworkPathRef + NetworkLinkRef

TransportLinkReference ➔ TransportLinkRef

DeletionDate ➔ HistoricDate (see note on dates)

QualityStatementId ➔ QualityRef

Day_TypeID ➔ DayTypeRef

TrafficEvent / ***LocationDesc ➔ LocationDescription***

Speciufc Attributes

AccessControl / DefaultState ➔ ***AccessControl DefaultStateTypeRef***

AccessControl / ***State ➔ AccessControl AccessControlStateTypeRef***

Accident / EndDate ➔ AccidenElementt / EndTime

CarParkConfiguration/ DetectorSystemCodeNumber ➔ CarPark / DetectorRef

Cttv /VRM Counbt ➔ Cctv / VehicleRegistrationMarksCount

Etc See Spreadsheet, Column S

UTMCXML REPRESENTATION OF ASSOCIATIONS & AGGREGEGATIONS

UTMXXML takes a consistent approach to the representation of relationships from the model as flate XML . Relationships will be dereferenced for serialisation using either XML nesting or explicit reference, depending on the modelsemantics.

1. **Composition**. Aggregated composition (indicated by a black diamond in UM) child elements are usually embedded in line in the XML.
    a. Multiple Child elements are embedded in line within a wrapper tag, named after the relationship. Names are generally polymorphic, eg Configurations (Not TransportLinkConfigurations)

© Copyright Kizoom 2007-2009

For example *"TransportLink <>----dynamics -→ TransportLinkGpsDynamic"* is a composition that allows for multiple children and would be represented in an XML document as a <Dynamics> wrapper tag

```
<TransportLink ">
        <SystemCodeNumber>1234</SystemCodeNumber>
        <Dynamics>
                <TransportLinkGpsDynamic>
                        <LastUpdated>2001-12-17T09:30:47.0Z</LastUpdated>
                        <AverageSpeed>30.5</AverageSpeed>

                </TransportLinkGpsDynamic>
        </Dynamics>
```

b. Single Child elements are embedded directly without a wrapper relationship names,

For example *"AirQuality <>----has-→ BasicData"* is a simple child. Wrapper tags are also used for composite data types such as *Point*.

```
        <AirQualityDefinition>
                <ObjectTypeRef>67</ObjectTypeRef>
                <QualityRef>45</QualityRef>
                <CreationDate>2001-12-17T09:30:47.0Z</CreationDate>
                <DataSourceTypeRef>0</DataSourceTypeRef>
                <Point>
                        <Easting>999999</Easting>
                        <Northing>1999999</Northing>
                </Point>
                <BasicData>
                        <MeasurementUnits>Carats</MeasurementUnits>
                </BasicData>
        </AirQualityDefinition>
```

2. **Associations**: Associations are implemented as a reference to the identifier i.e. foreign key of the associated element. This is named so as to indicate it a reference and not a value.

   a. Single multiplicity associations (0:1 or 1:1) use xxRef for foreign keys that implement associations are indicated by xxxRef . These will be named polymorphic for generic relationships (eg to ObjectType ➔ ObjectTypeRef) , and specifically for non generic relationships eg DataSourceTypeRef

```
<TransportLinkDefinition>
                <SystemCodeNumber>1234</SystemCodeNumber>
                <ObjectTypeRef>SCOOT</ObjectTypeRef>
                <QualityRef>Qual01</QualityRef>
                <CreationDate>2001-12-17T09:30:47.0Z</CreationDate>
                <DataSourceTypeRef>QMISS</DataSourceTypeRef>
                <NetworkPathRef>Net001</NetworkPathRef>
</TransportLinkDefinition>
```

   b. Multiple multiplicity associations (0:* or 1:*) use a wrapper tag for relationships. For example a network path may reference multiple NetworkLinks

```
        NetworkPath>
                <SystemCodeNumber>NTTN3434</SystemCodeNumber>
                <CreationDate>2001-12-17T09:30:47.0Z</CreationDate>
                <NetworkPathTypeRef>23</NetworkPathTypeRef>
                <Permanent>true</Permanent>
                <InitialTrim>23</InitialTrim>
                <FinalTrim>2222</FinalTrim>
                <Links>
                        <NetworkPathListEntry>
                                <OrderId>1</OrderId>
```

© Copyright Kizoom 2007-2009

```
                    <NetworkLinkRef>Ntl2345</NetworkLinkRef>
                </NetworkPathListEntry>
                <NetworkPathListEntry>
                    <OrderId>2</OrderId>
                    <NetworkLinkRef>Ntl2347</NetworkLinkRef>
                </NetworkPathListEntry>
                <NetworkPathListEntry>
                    <OrderId>3</OrderId>
                    <NetworkLinkRef>Ntl2349</NetworkLinkRef>
                </NetworkPathListEntry>
            </Links>
        </NetworkPath>
```

UTMCXML DATA TYPES

A consistent set of Simple XML Data Types are used

a. Base types e.g. boolean, integer, dateTime, duration

b. Specific dimensioned types e.g. speed, vehiclePerInterval, etc. See

c. Identifier types: these are generally strongly typed eg for the SystemCodeNumber of TransportLink we use TransportLinkId, not ObjectId

Group	Type	Base	
xml	Xsd:string		Used for all text. Should this be NL STRING with LANG?
	Xsd:boolean		
	Xsd:dateTime		Used for all timestamps Normally in UTC
	Xsd:duration		Used for all time intervals.
	Xsd:url		
Group	Simple type	Base type	Note
UTMC Identifiers	ObjectIdType	Xsd:NMTOKEN	Used for SystemCodeNumber, restricted on each
	ObjectTypeIdType	Xsd:string	Used for all xxxTypeDefinitions restricted by subtype
	LogIdType	Xsd:integer	Used for all xxxDeviceHistory restricted by subtype
	FaultIdType	Xsd:integer	Used for all xxxFault restricted by subtype

© Copyright Kizoom 2007-2009

	CommandIdType	Xsd:integer	
	DayTypeIdType	Xsd:integer	
	QualityIdType	TypeIdType	Used for all xxxFault restricted by subtype
UTMC Dimensioned	speed	Xsd:floar	
	distance	Xsd:decimal	
	amount	Xsd:decimal	monetary
	concentration	Xsd:decimal	
	temperature	Xsd:decimal	
	humidity	Xsd:decimal	
	pressure	Xsd:decimal	
	vehiclesPerInterval	Xsd:decimal	
	vehiclesPerMinute	Xsd:decimal	
	numberOfVehicles	Xsd:integer	
	laneOccupancy	percentage	
	normalizedString	Xsd:string	
	numberOfPixels	Xsd:integer	
	percentage	Xsd:float	
	colour	Xsd:nmtoken	
Other	EmptyType	-	
	Extensions	Xsd:any	

© Copyright Kizoom 2007-2009

UTMCXML Coding For Maintainability

The XML schema is coded to facilitate reuse of the XML objects in many different applications:

1. Modularisation for maximum maintainability
    a. Use a modular package structure, with a strictly linear dependency graph.
    b. There is a distinct XML package for each UTMC Data Object, allowing it to be used (and maintained) independently of other objects, and for new objects to be added Overall there are packages that correspond to the UML packages – with some additional packages for XML and protocol plumbing.
    c. The representation of the model is distinct from the representation of the protocol, allowing the same model to be used in different protocols (e.g. publication, SIRI service requests, etc).
    d. Substitutable types are used in the framework to indicate where new data objects can be added simply by adding a new package and without changes to the rest of the schema.
    e. UtmcXml packages are self-contained with no external dependencies , and are all within a a single UTMC namespace.
    f. Include an Extensions placeholder to allow arbitrary extension.
2. Modularisation for maximum reuse
    a. Use of additional abstract XML supertypes where appropriate.
    b. Use of Groups where appropriate.
    c. Include identifiers (e.g. ***SystemCodeNumber***) on individual subcomponents of Data Objects so that they can be used standalone or embedded within a wider context.
3. Use systematic versioning
    a. Packages should be versioned hierarchically, so that if an included package changes, so does the parent.
    b. Version attributes should be included on objects to permit reflection by applications.
4. Code to assist simplicity in the implementation.
    a. In some places implementation is simplified by the use of explicit choice rather than substitution group references.

UTMCXML SCHEMA PACKAGEGE STRUCTURE

UTMCXML SCHEMA PACKAGEGE OVERVIEW

The UTMC schemas are made up of reusable modules, shown in outline in Figure 2-1.

© Copyright Kizoom 2007-2009

- The core UTMC model (corresponding to the current UTMC UML) is defined by a UTMC model package that assembles the different groups of UTMC objects. This is made up of multiple packages, one per data object each based on a common UTMC Abstract Object Model. Data Objects of a given type (e.g. Devices, Traffic Events) are further grouped so they can be referenced together - see below. Network objects are defined in a simple package.

- A set of Filter packages reference their respective model data objects, e.g. device, network

- The UTMC SIRI- Service schema references the UTMC Model, UTMC Filters and the SIRI framework

- The UTMC **Publication schema references** the UTMC Model & UTMC Filters packages

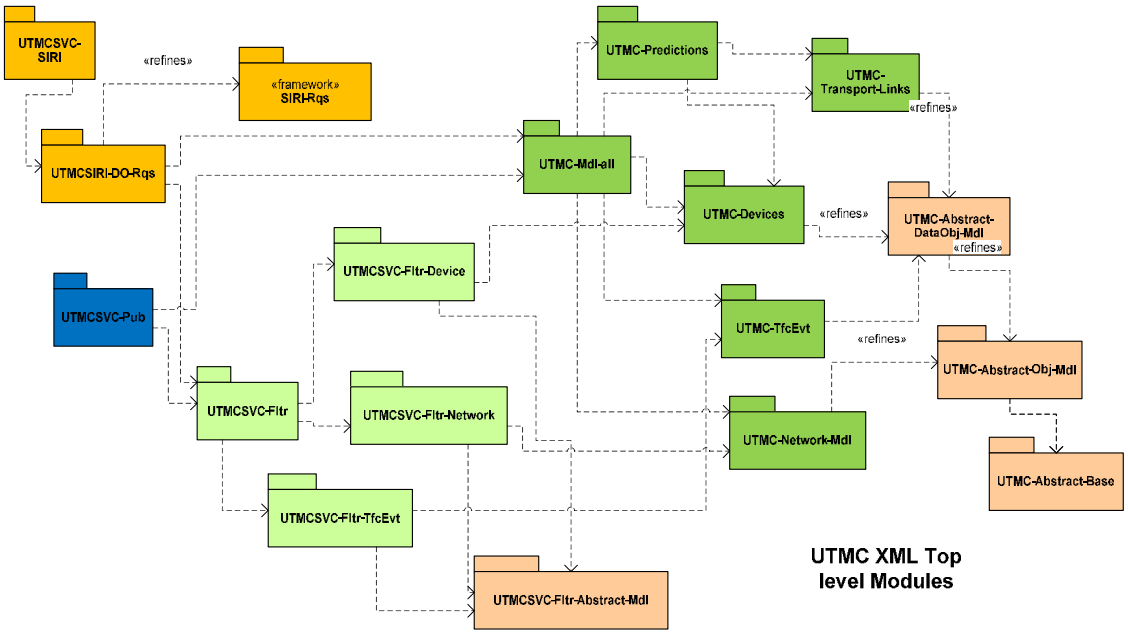


Figure 2-1 UTMC High level Modules

© Copyright Kizoom 2007-2009

UTMC XML SCHEMA MODEL PACKAGES

Figure 2-2 shows dependencies between UTMC Data Object packages: certain Data Objects reference other objects, for example Transport Routes Reference Transport Links

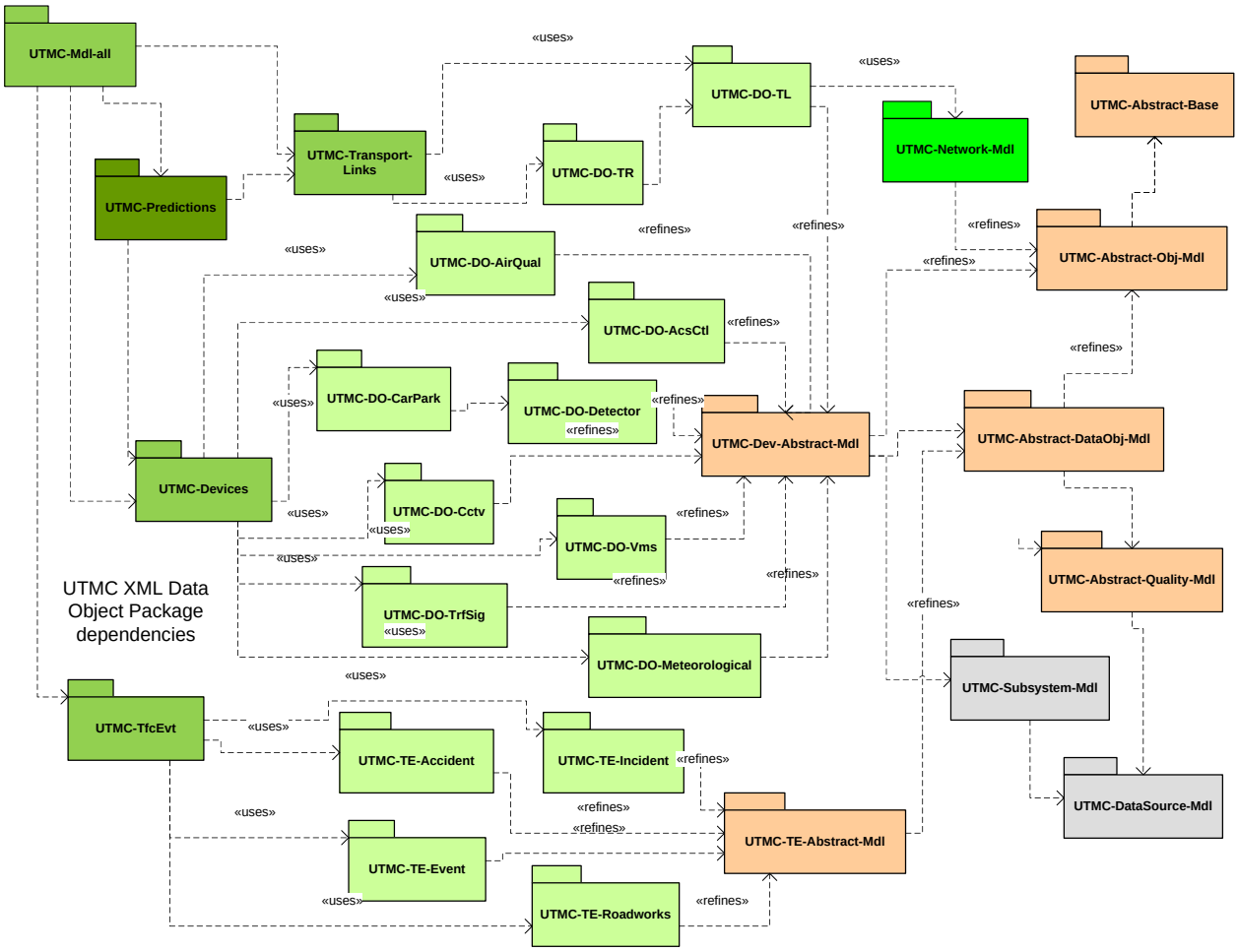


Figure 2-2 UTMC Model packages

UTMC XML SCHEMA ABSTRACT COMPONENT PACKAGES

The UTMC Data objects are built as extensions of a set of framework packages

© Copyright Kizoom 2007-2009

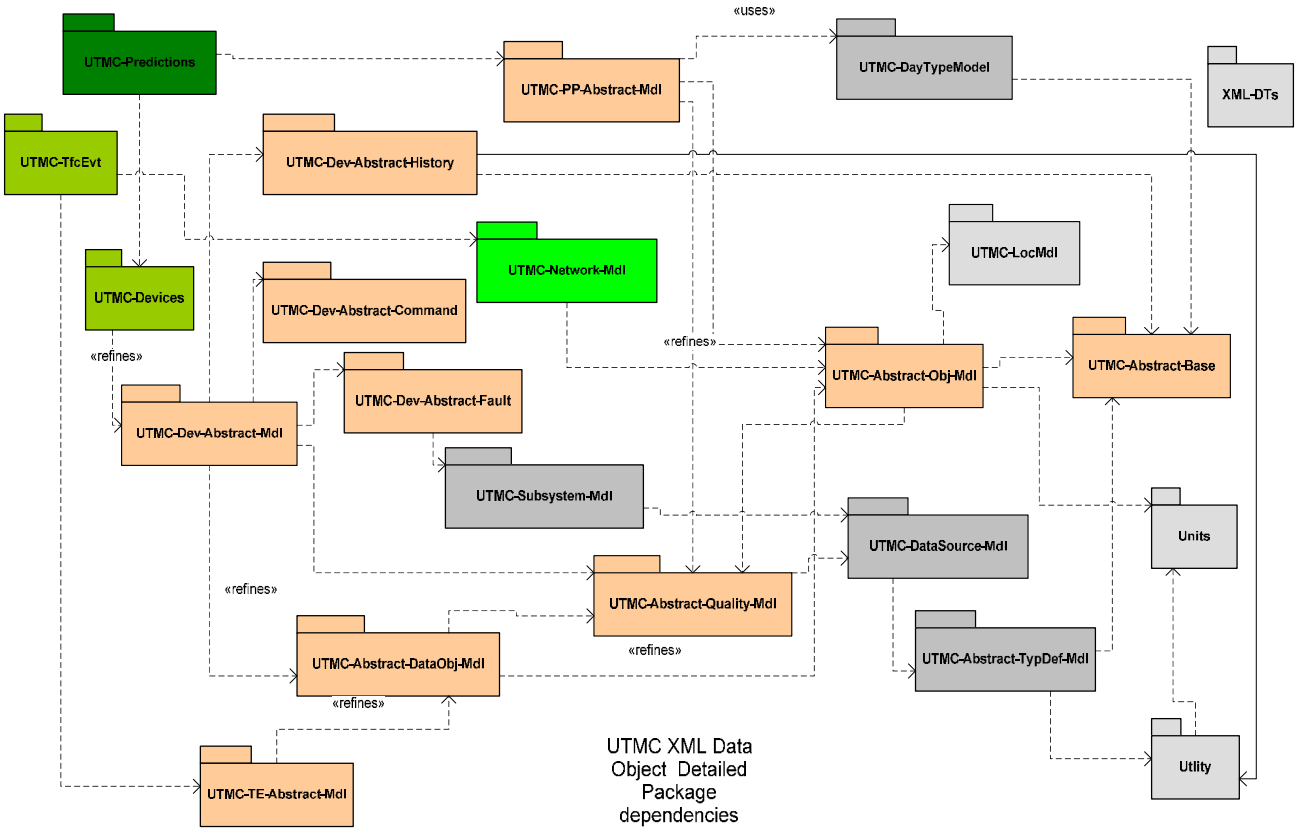
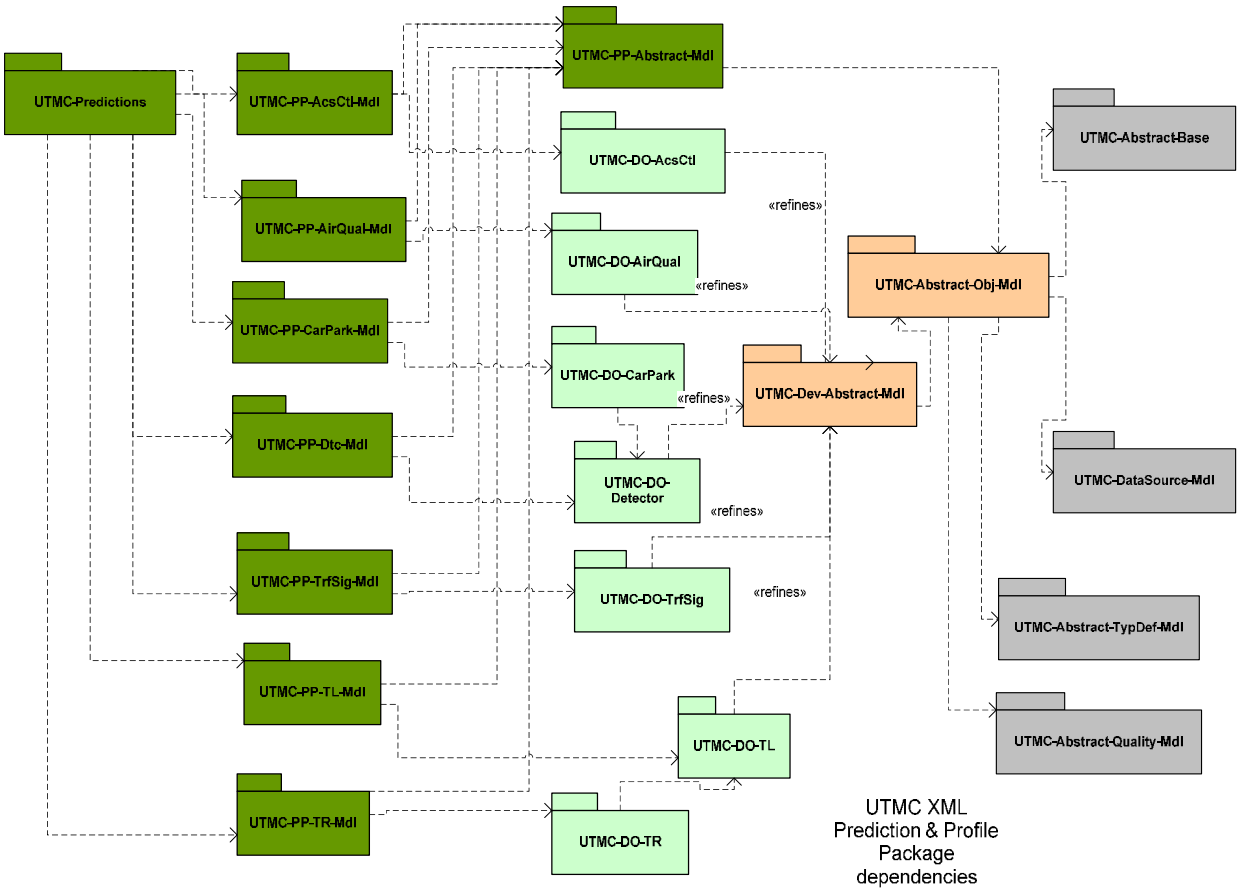


Figure 2-3 Detailed ABstarct and component pacjages

UTMCXML SCHEMA PREDICTION & PROFILE PACKAGES

a. The Prediction & Profile packages are built as extensions of the bas data object packages

© Copyright Kizoom 2007-2009



UTMCXML DOCUMENTATION

The UtmcXml will provide the interface to which many application programmers work, it is intended to be fully documented according to the following principles.

2. Code for maximum readability
    a. Include documentation on every element making the schema self documenting as far as possible...
    b. Use groups to indicate the relationship of elements to complex types, and to reflect and meaning semantic groups.
    c. Order elements in a consistent semantic order, eg identifiers, classifiers, children
    d. Use Substitution groups to shown the object hierarchy and the allowed values.
3. Code egif /Govtalk documentation standards
    a. Include documentation headers with GMS metadata, IPR, change notes etc etc

© Copyright Kizoom 2007-2009

3. UtmcXml extensions to UTMC

XML makes it easy to have optional elements in a model that can be omitted if not required. In developing the UtmcXml to be as "future proof" as possible, a number of minor enhancements have been included. These fall into two groups;

- **Minor enhancements** that have already been identified from current UTMC use (e.g. support for alias identifiers). In some cases these are just a more specific typing of the existing model.
- **Enhancements to support Freeflow**. These include both additional Data Objects (which do not affect other objects) and a small number of enhancements to support generic features such as historic data.

UTMCXML MINOR ENHANCEMENTS

The following is a list of minor enhancements

1. Clarification of NetworkPathReference. The UTMC Network path reference may be used to reference either a ***NetworkPath*** object or a ***NetworkLink*** Object. This is made explicit in the UtmcXML schema so that consumer systems know which type of object is referenced
2. Data Object Identifier **Alias Support:** Some systems have to rename objects on import. Support for an originating identifiers makes it possible to retain identity on round trip or repeated exchange. An ***OriginatingObjectRef*** of the same type as the ***SystemCodeNumber*** has been added
3. Explicit System uniqueness: All data objects can indicate a unique system identifier within which they are guaranteed to be unique (The ***Participant*** Identifier) . In many cases this will correspond to the UTMC **Data Source**; in other cases it may correspond to the UTMC **subsystem**. The *ParticipantId* provides a uniform mechanism that can be used to manage system identity.
4. Type Correctness of Device History/ Log Identifiers:
    - Use of a Type specific identifier for Log entries (Since these are unique with object Type)
    - Use of explicitly dimensioned data type e.g. *speed, percentage* etc
5. External incident references
    - To support round trip identity management when exchanging incidents with external systems there is support for external identifiers on UtmcXml Traffic events. This makes it possible to associate an external reference with each external reference.
6. GIs Feature/TOID References.
    - As well as Point Coordinates, NetworkGeometry elements may also include an explicitly reference to a GIS system feature reference such as a TOID for a Road link

© Copyright Kizoom 2007-2009

7. Prediction & Profile Objects ; These are not well specified in the UTMC Specification – which suggests the use of a meta data mapping: eg *"attribute x = value b".*

    - In the XML implementation we propose using the explicit values, simplyreusing the dynamic value definitions from each object type. Predictions are not provided for Meteorological (which ahs a forecast flag) or VMS.

8. Presentation Emphasis support: Colour, Iicons

    - The ability to optionally associate colour and icons is supported. ***Colour*** & ***Image olink*** support is on to Abstract Type Definition, Abstract Dynamic as a simple attribute. This allows a default colour to be associated with . The base state and different dynamic states. Colours can also be associated with

9. Test Data Reality

    - Add optional reality attribute based on Datex2 value so that data can be flagged as test , live, emergency service exercise , etc. this allows the same infrastructure to be used for test and live data.

10. MultiLine Vms

    - Allow specification of number of lines of VMS

UTMCXML FREEFLOW ENHANCEMENTS

There are some enhancements that specifically assist Freeflow implementations

1. Support for Historic Data / Use of Wrappers

    a. Use of a "Wrapper" Data object so that all component elements of a UTMC, including Definitions can have multiple instances – see below

    b. Addition of a lasted Updated / and deletion / historic attributes to all data objects so that historic data can be supported – See below.

    c. external systems there is support for external identifiers on traffic events

2. Additional Object types: Two draft Freeflow data objects have been added as separate packages

    a. Alert

    b. Intervention

DATA OBJECT WRAPPER

The UtmcXml extends the UTMC model to support full historic data. This is reflected in the use of wrapper objects as described below.

UTMC DATABASE MODEL FOR UTMC DATA OBJECT STRUCTURE

© Copyright Kizoom 2007-2009

The UTMC data base model has Configuration and Dynamic records as composite children of the Data Object Definition object, distinguished by their creation date.

xxxDefinition (PK=SystemCodeNumber)
xxxConfiguration (PK=SystemCodeNumber, PK=ConfigurationDate)
xxxDynamic (PK=SystemCodeNumber, PK=LastUpdated)
etc

UTMC UML "CLASSIC" MODEL FOR UTMC DATA OBJECT STRUCTURE

The UTMC UML model interprets Configuration and Dynamic elements as children of a Data Object Definition, suggesting the following composition.

xxxDefinition (PK=SystemCodeNumber)
-0:* ***xxxConfiguration*** (PK=SystemCodeNumber, PK=ConfigurationDate)
-0:* ***xxxDynamic*** (PK=SystemCodeNumber, PK=LastUpdated)
-0:* etc

UTMC XML MODEL FOR UTMC DATA OBJECT STRUCTURE

However rather than using a simple nesting as above, the UtmcXml instead introduces an additional wrapper data object. This is a slight generalization the UML model representation that makes it possible to have multiple historic definitions. The interpretation does not contradict the database/UML representation in that there is still a 1:* derivable relationship between Definition and Configuration/Dynamic elements).

xxxDataObject (PK=SystemCodeNumber)
-1:* ***xxxDefinition*** (PK=SystemCodeNumber, LastUpdated)
-0:* ***xxxConfiguration*** (PK=SystemCodeNumber, ConfigurationDate)
-0:* ***xxxDynamic*** (PK=SystemCodeNumber, LastUpdated,)
-0:* etc

The UTMC model also further supports the use of two dates on all objects – a Last Updated/Configuration date – the date and time of change, and a Historic Date – the date and time at which a record was superseded by a subsequent update. Systems that don't support historic data can use the dates as at present. Systems that do can use the the Historic date (which will be blank for current records) as a simple unique identifier of records within data object instance.

xxxDataObject (PK=SystemCodeNumber)
-1:* ***xxxDefinition*** (PK=SystemCodeNumber, LastUpdated, PK= HistoricDate)
-0:* ***xxxConfiguration*** (PK=SystemCodeNumber, ConfigurationDate. PK= HistoricDate)
-0:* ***xxxDynamic*** (PK=SystemCodeNumber, LastUpdated, PK= HistoricDate)
-0:* etc

© Copyright Kizoom 2007-2009

4. THE UTMC XML REPRESENTATION

The UTMC XML representation is described by the UtmcXml XML Schemas (These can be convenient accessed though utmc_publication.,xsd, and utmc.xsd) . The following diagrams provide an overview of the representation, indicating the names of elements and attributes – this can be used to understand and verify the implementation of the definitive UTMC UML as XML. Colour is used to indicate abstract and functional groups of classes. The primary identifiers are indicated by <PK>. References are indicated by <FK>

ABSTRACT DATA OBJECT – DETAILS

All data objects share some common properties. These are factored out as abstract classes.

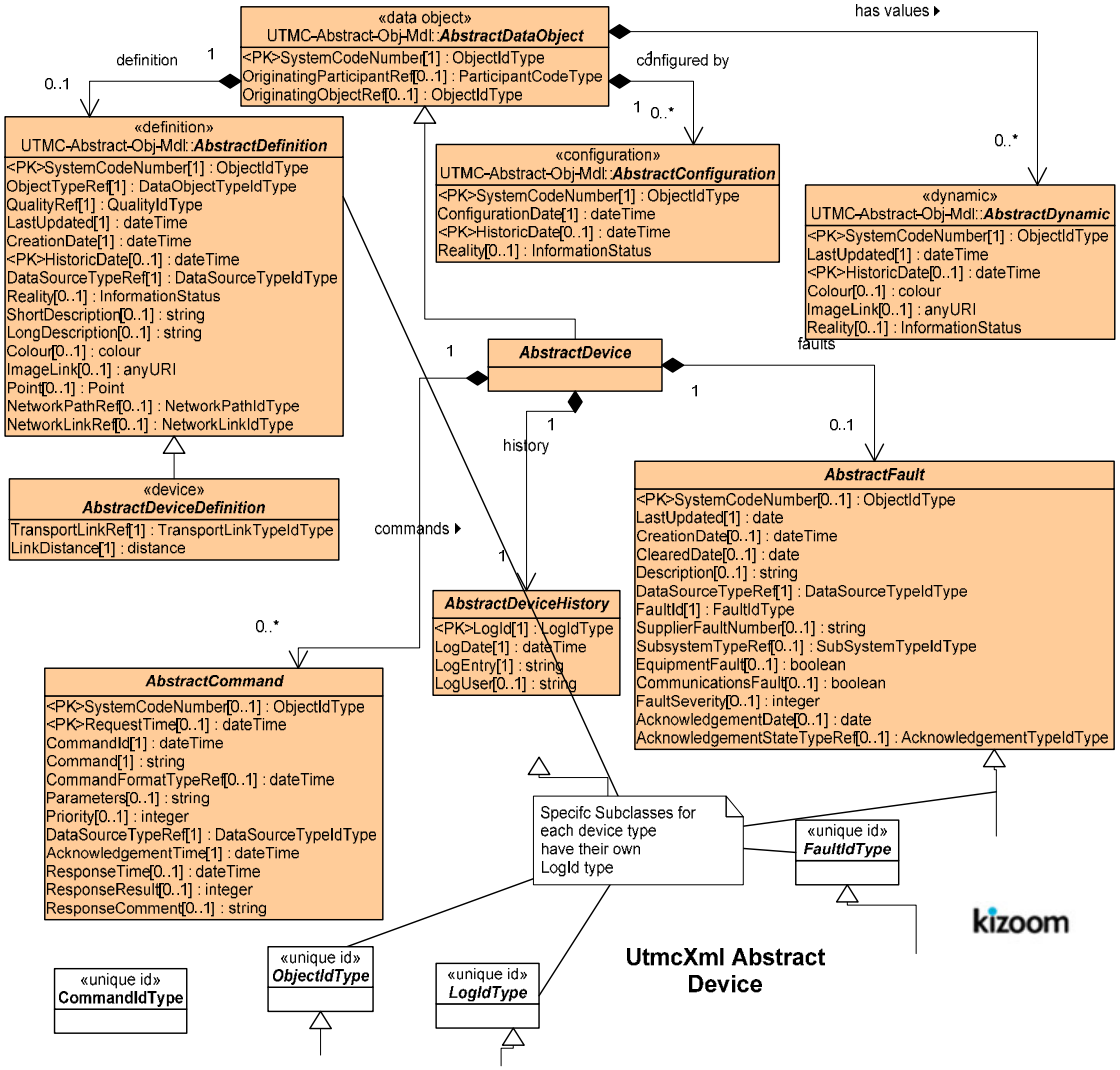


Figure 4-1 UML diagram of UTMC Abstract Data Object

© Copyright Kizoom 2007-2009

DATA OBJECT TYPE DEFINITIONS HIERARCHY

UTMC does not use many fixed enumerations to indicate the allowed values of elements. Instead it makes widespread use of a Type Definition object to define all codes , including (i) Data Object subtypes, (ii) specific states allowed fro particular objects, e.g. TrafficEvent Severity, LinkStatus, etc .

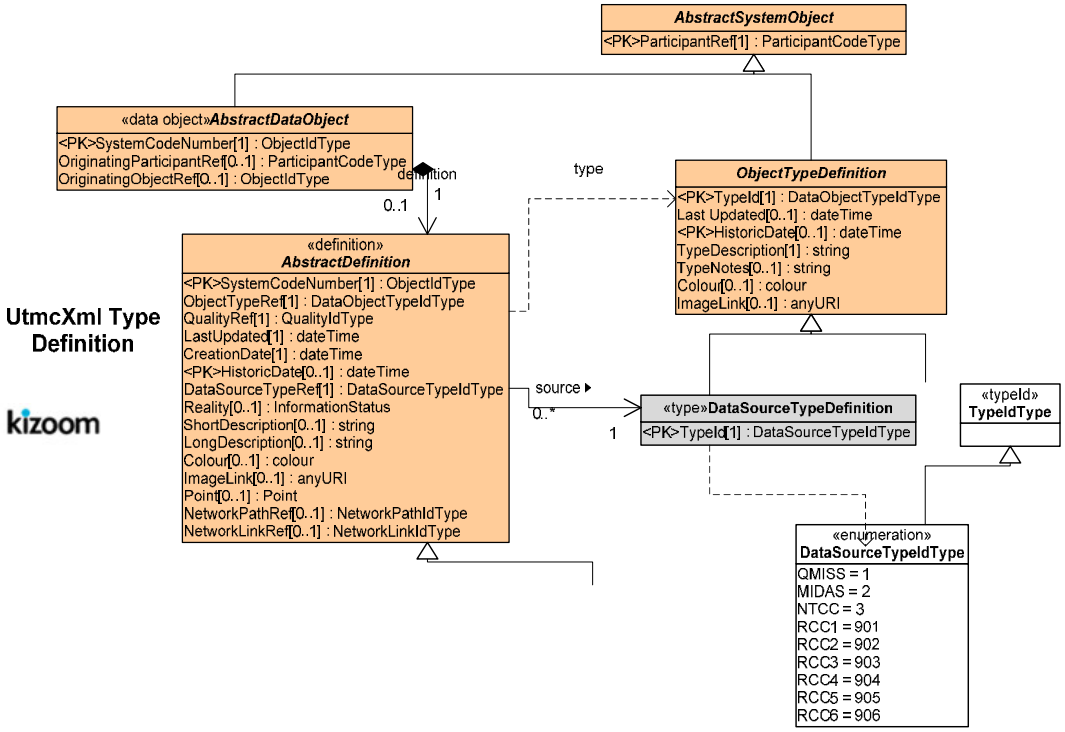


Figure 4-2 UML diagram of Abstract Object type

© Copyright Kizoom 2007-2009

Object and other Type definitions based on the Type definition are shown in Figure 4-4 UML diagram of Type Definitions.

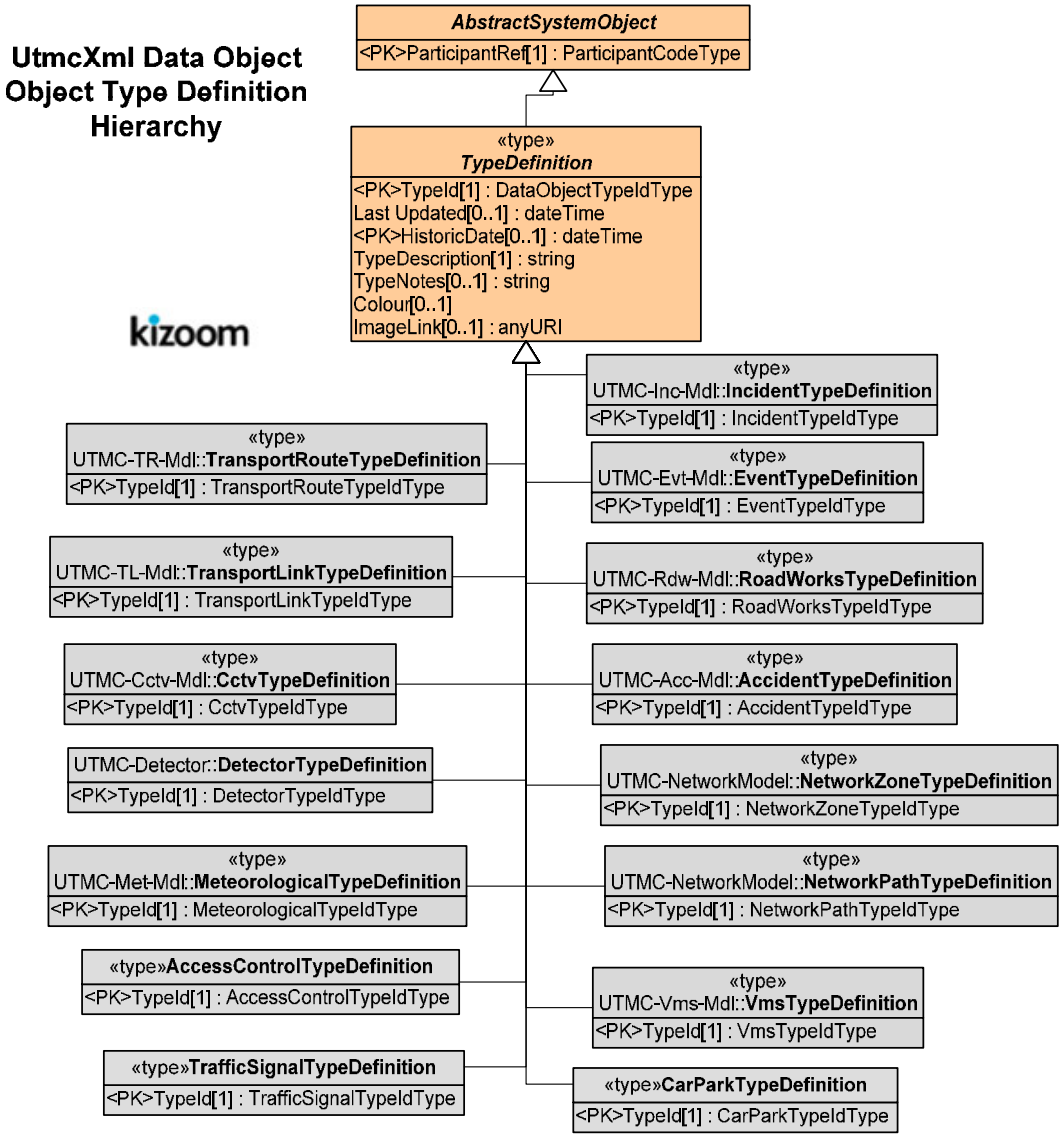


Figure 4-3 UML diagram of Type Definitions - Object Types

© Copyright Kizoom 2007-2009

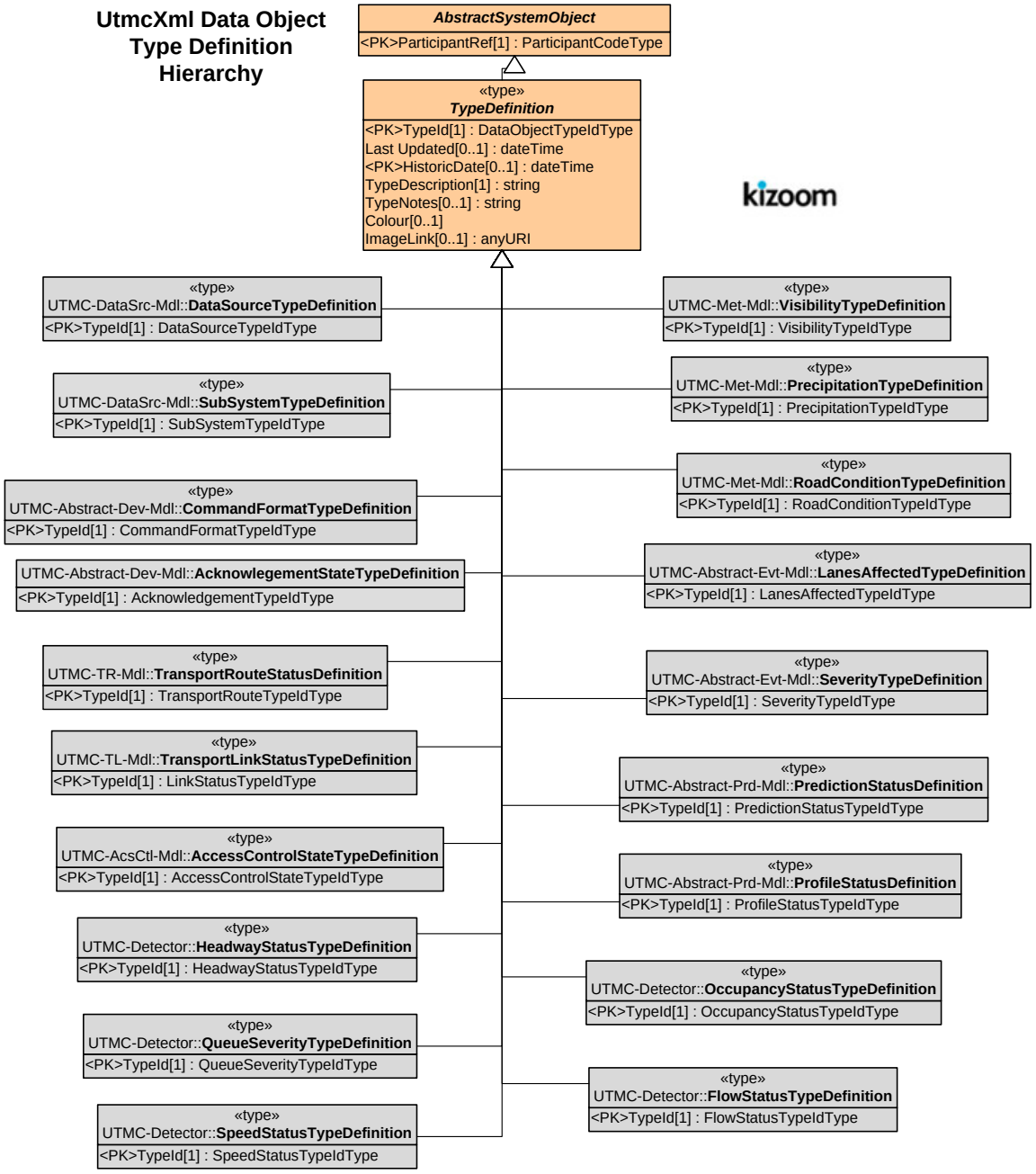


Figure 4-4 UML diagram of Type Definitions - Other

© Copyright Kizoom 2007-2009

CONCRETE DATA OBJECT COMPONENTS

Figure 4-5 shows the hierarchy of UTMC wrapper data objects

DATA OBJECT HIERARCHY

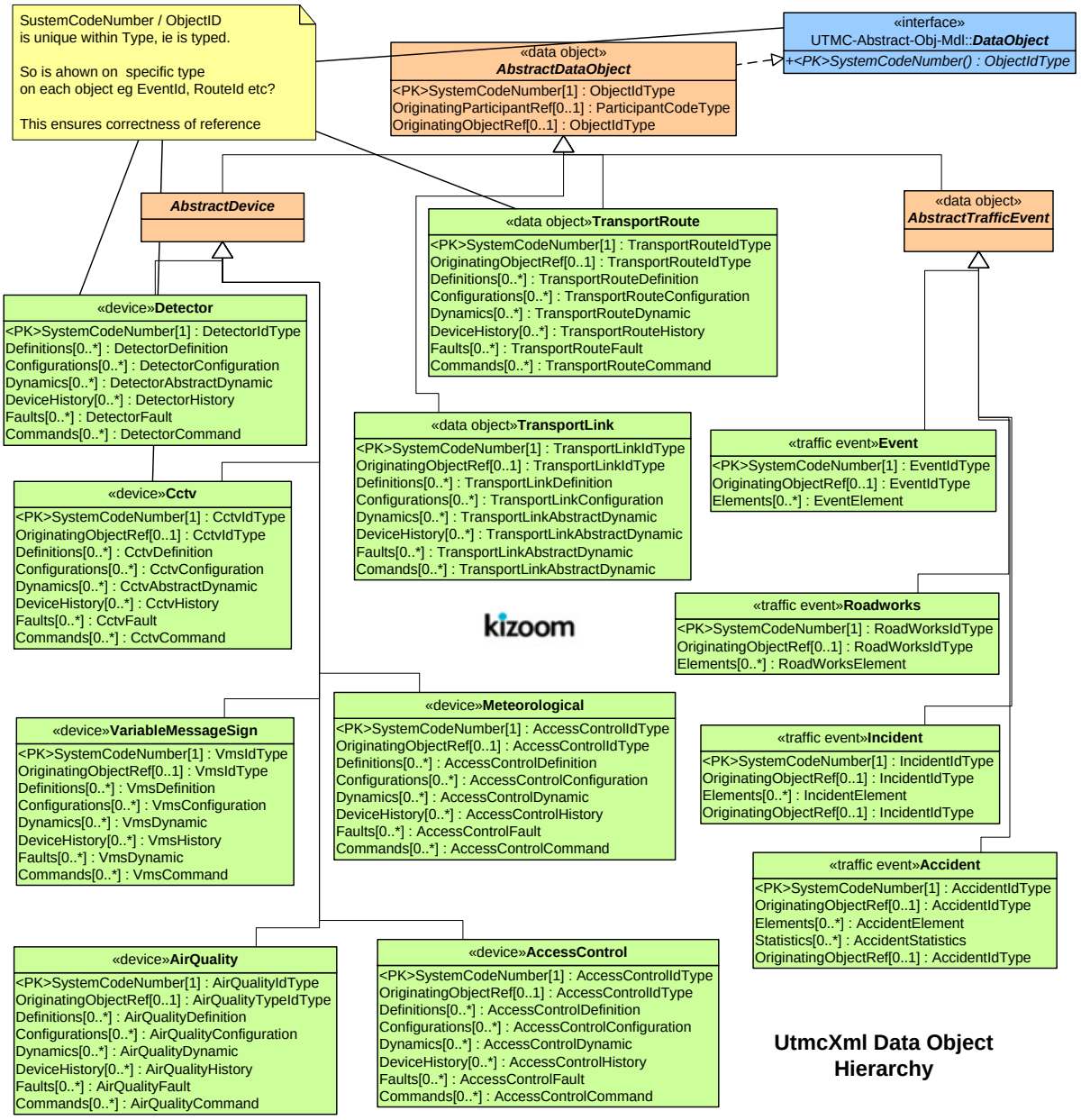


Figure 4-5 UML Diagram of UTMC Wrapper Data Objects

© Copyright Kizoom 2007-2009

DATA OBJECT DEFINITIONS HIERARCHY

Each UTMC data object can have a definition element.

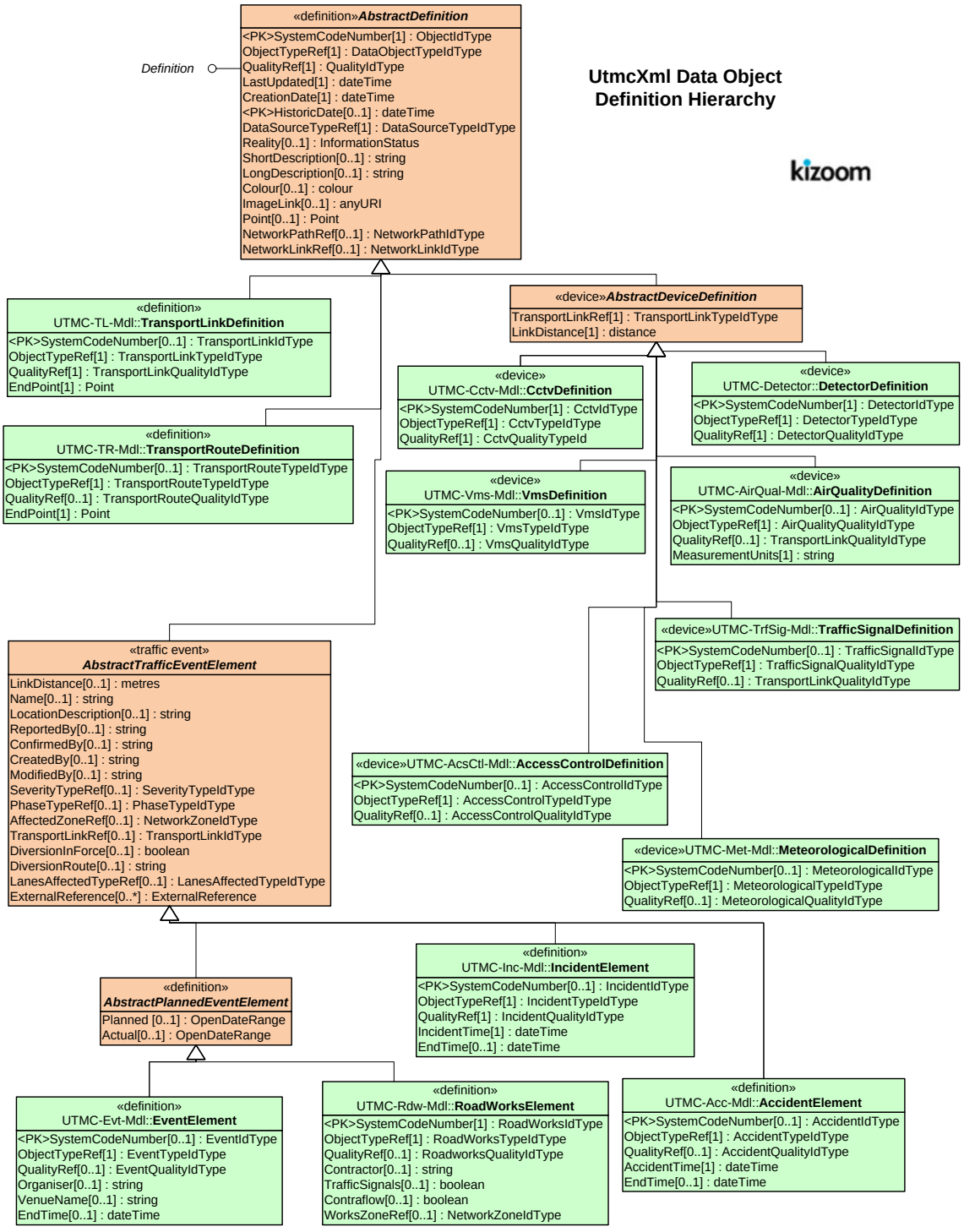
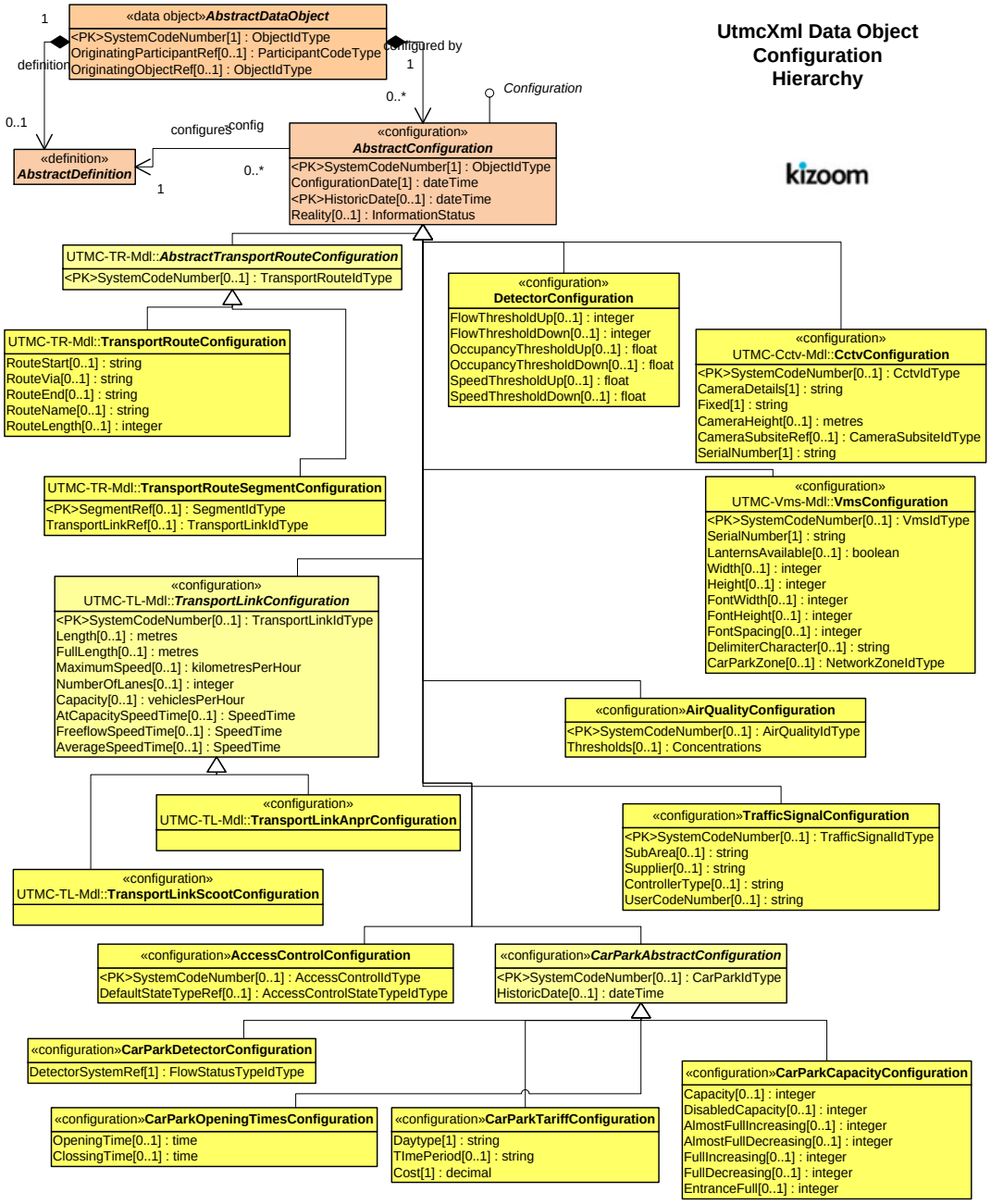


Figure 4-6 UML diagram of UTMC Data Object Hierarchy

© Copyright Kizoom 2007-2009

DATA OBJECT CONFIGURATIONS HIERARCHY

Device and Transport Link data objects can have a configuration element or elements depending on their type.



© Copyright Kizoom 2007-2009

DATA OBJECT DYNAMICS HIERARCHY

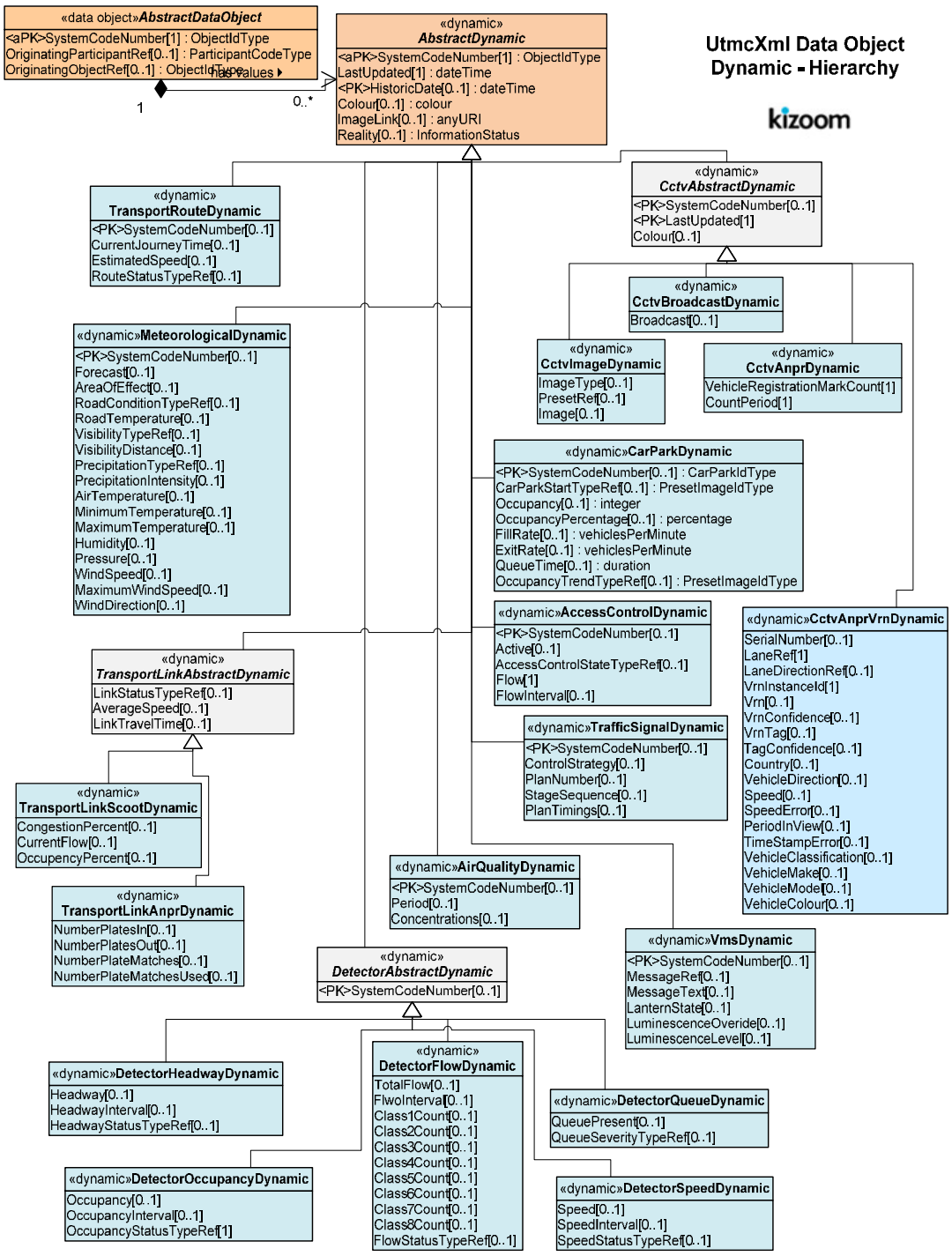


Figure 4-7 Data Object Dynamic Hierarchy

DATA OBJECT QUALITIES HIERARCHY

© Copyright Kizoom 2007-2009

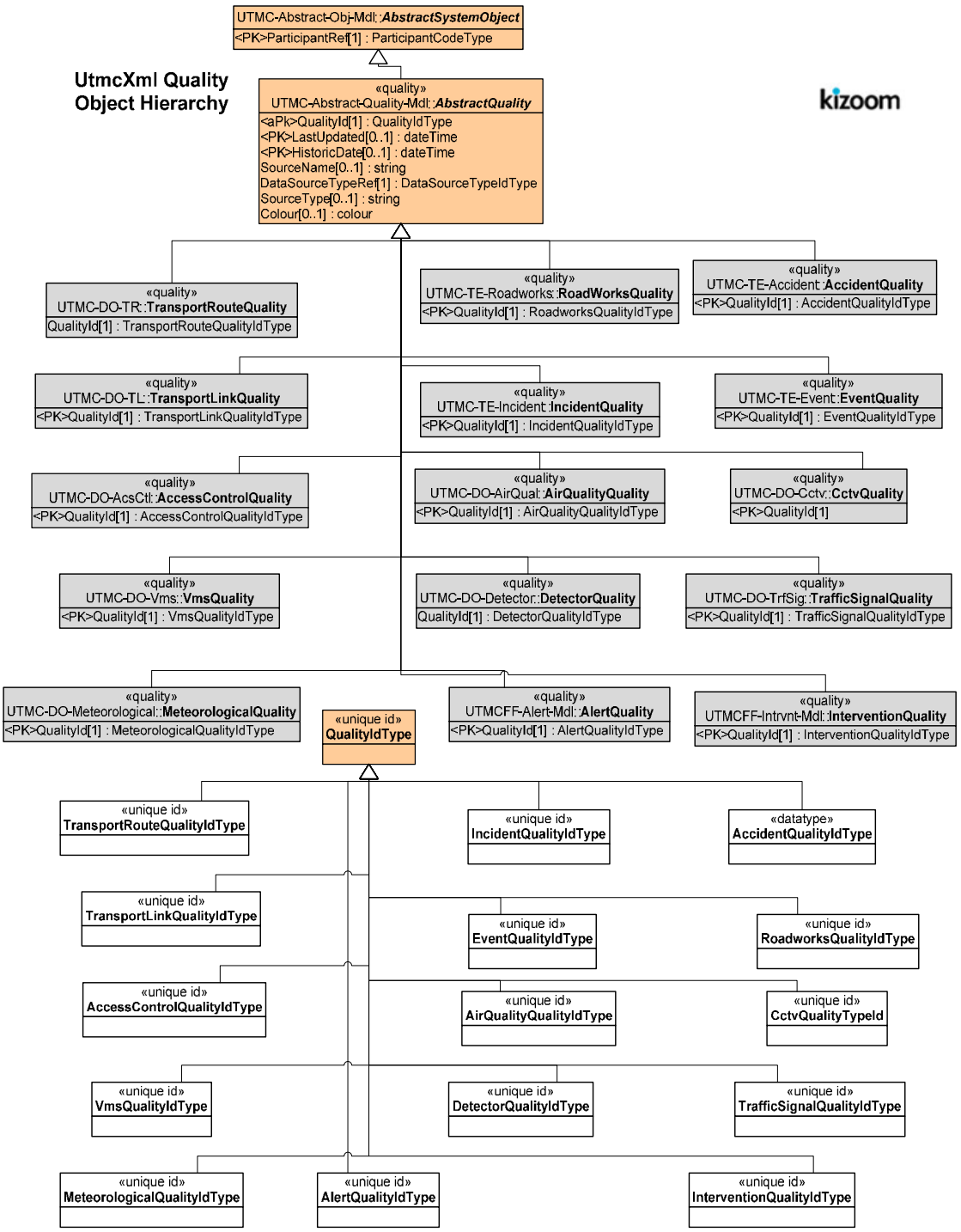


Figure 4-8 UML diagram of UTMC Quality Objects

© Copyright Kizoom 2007-2009

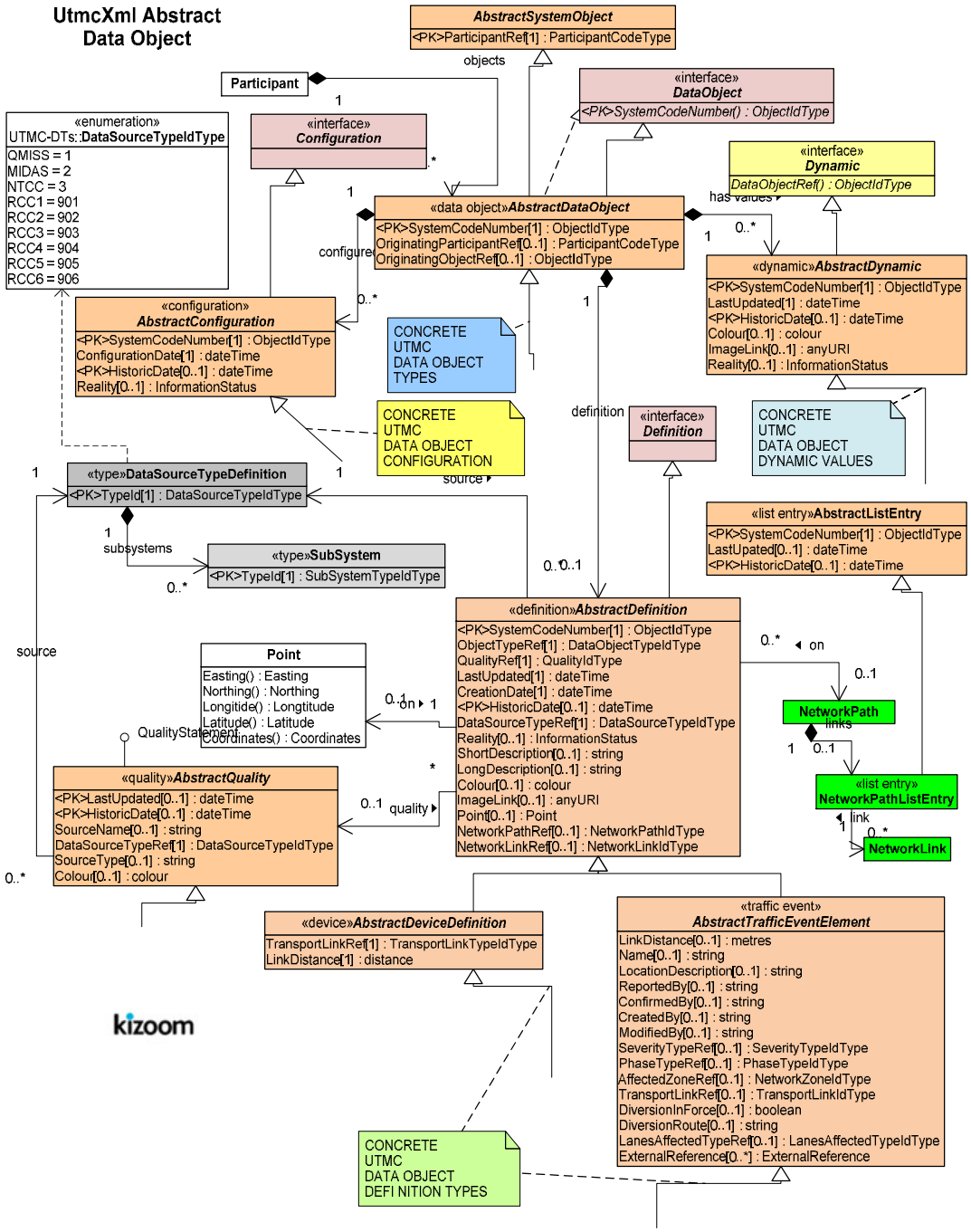


Figure 4-9 UML diagram of UTMC Abstract Object

© Copyright Kizoom 2007-2009

ABSTRACT DEVICE OBJECT

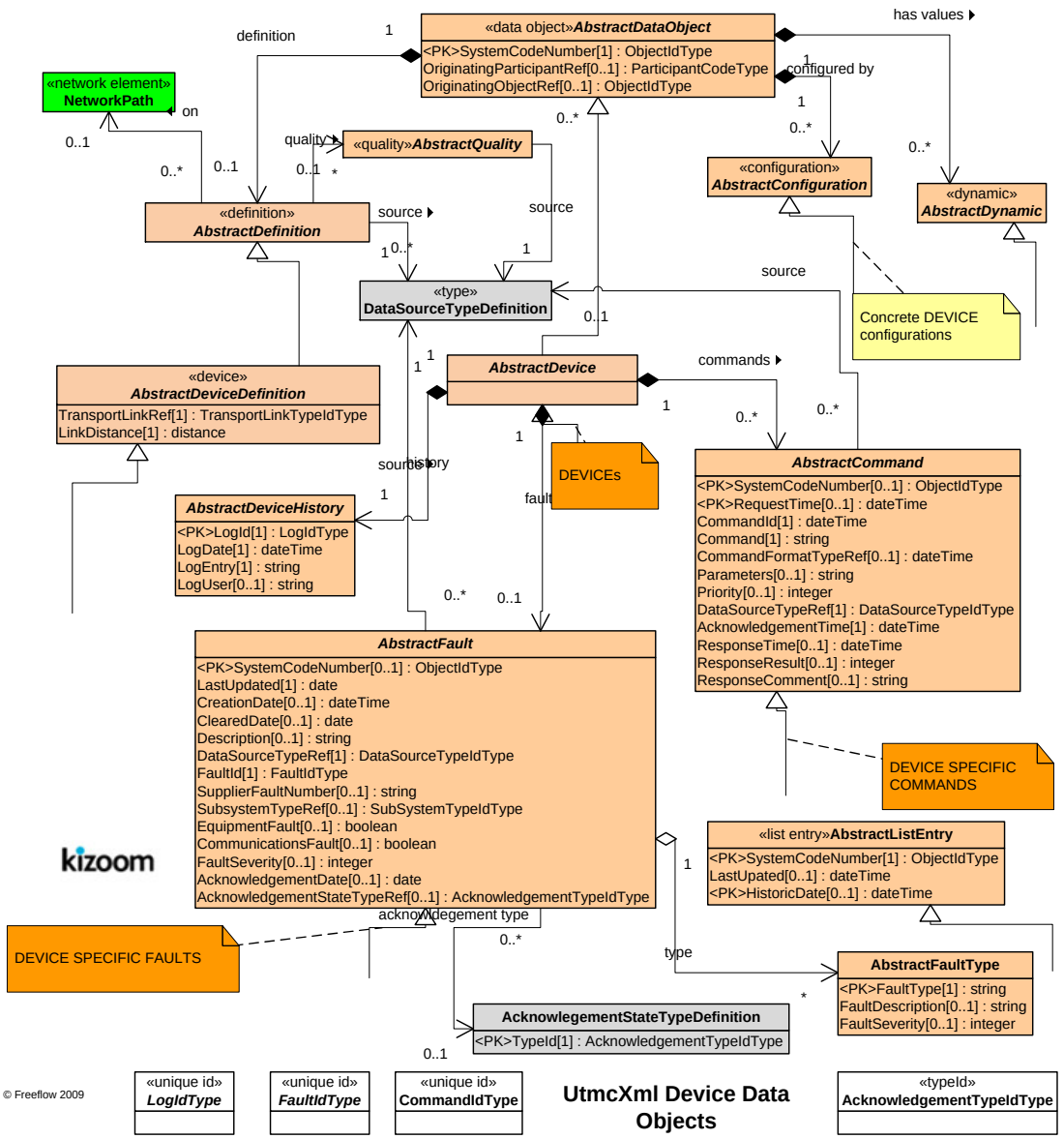
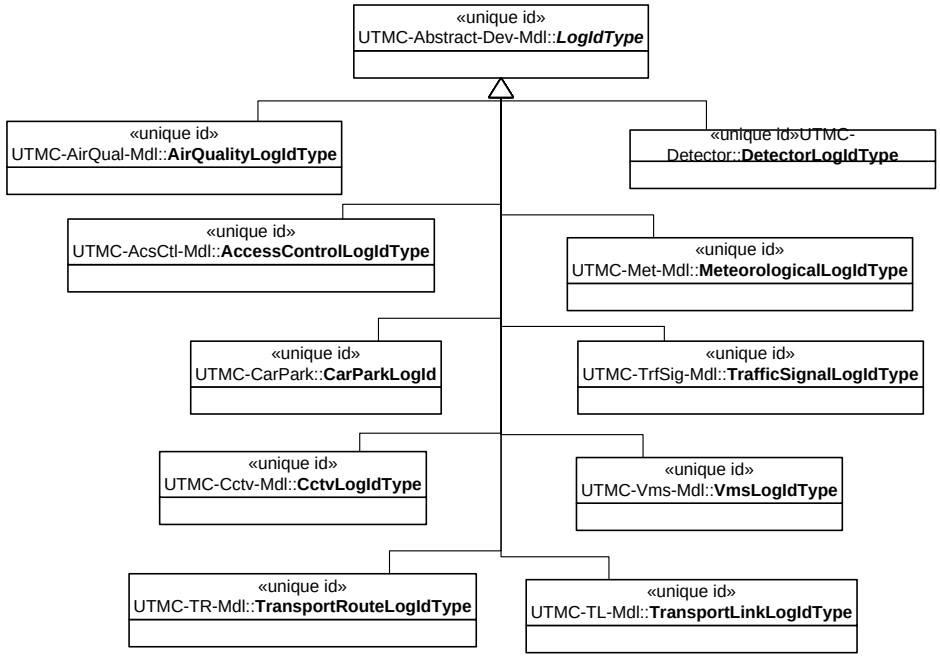


Figure 4-10 UML diagram of UTMC Abstract Device Object

© Copyright Kizoom 2007-2009

DEVICE HISTORY HIERARCHY



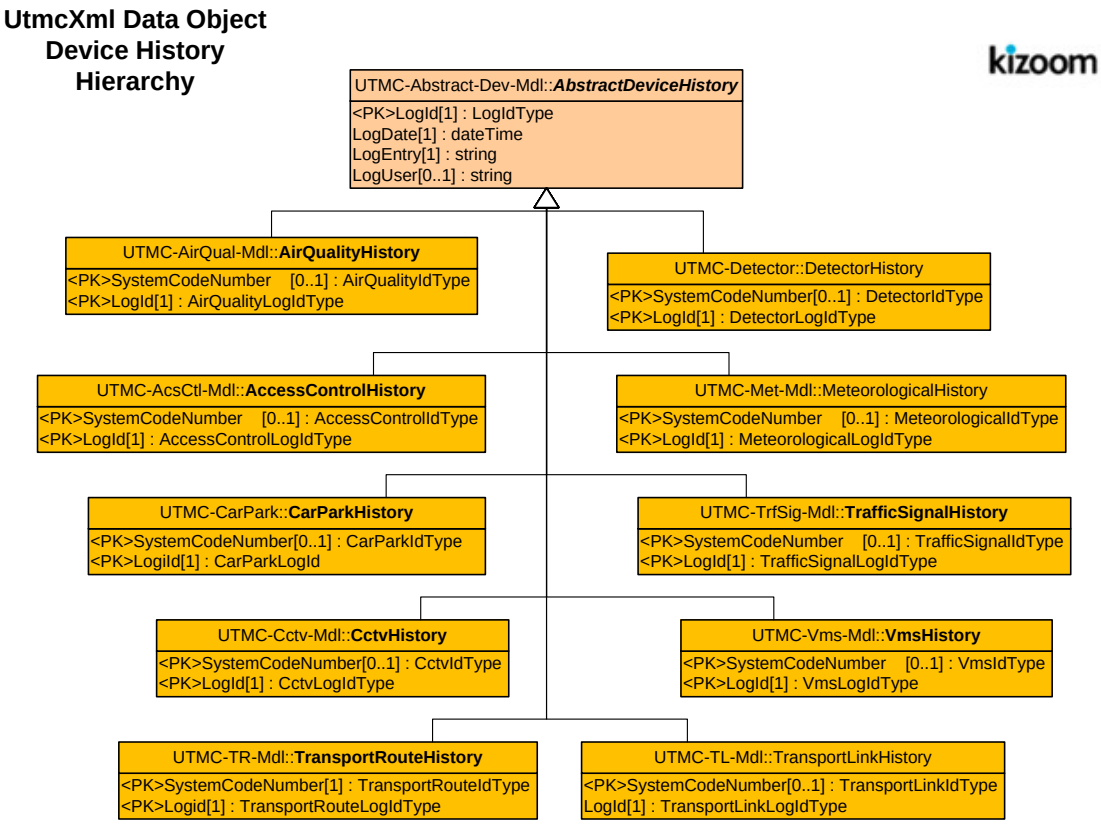


Figure 4-11 UML diagram of UTMC Device History Hierarchy

© Copyright Kizoom 2007-2009

DEVICE FAULT HIERARCHY

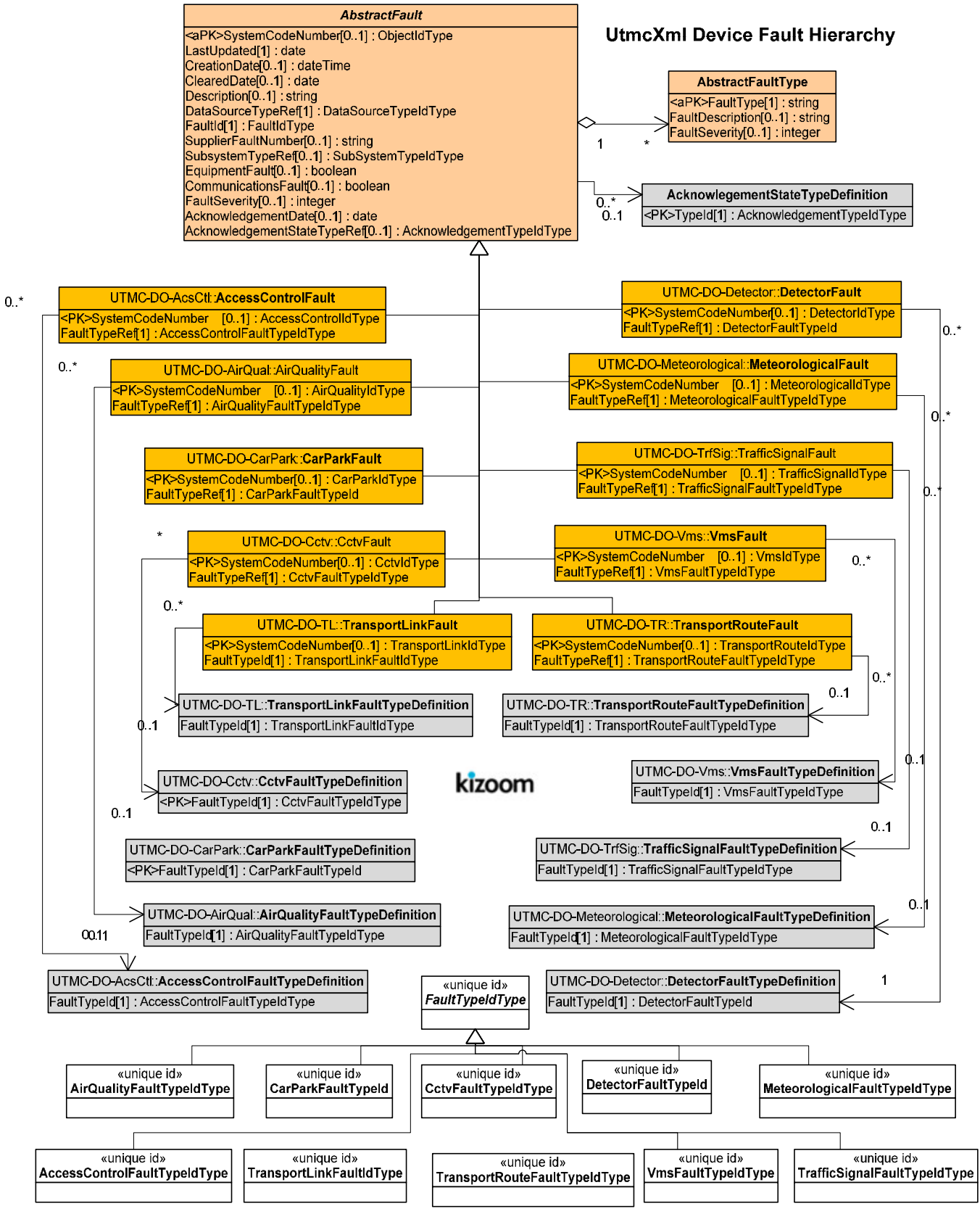


Figure 4-12 UML diagram of UTMC Device Fault Hierarchy

© Copyright Kizoom 2007-2009

© Copyright Kizoom 2007-2009

DEVICE COMMAND HIERARCHY

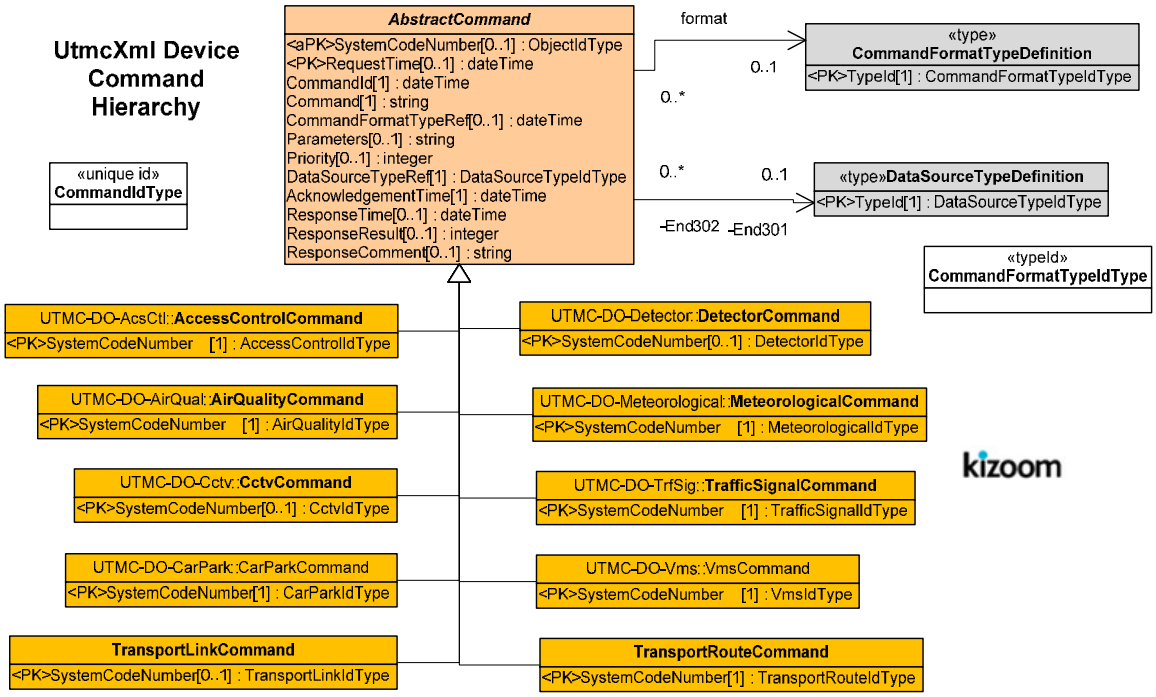


Figure 4-13 UML diagram of UTMC Device Command Hierarchy

© Copyright Kizoom 2007-2009

5. CONCRETE UTMC DATA OBJECTS

UTMC NETWORK OBJECTS

UTMC NETWORK OBJECTS – OVERVIEW

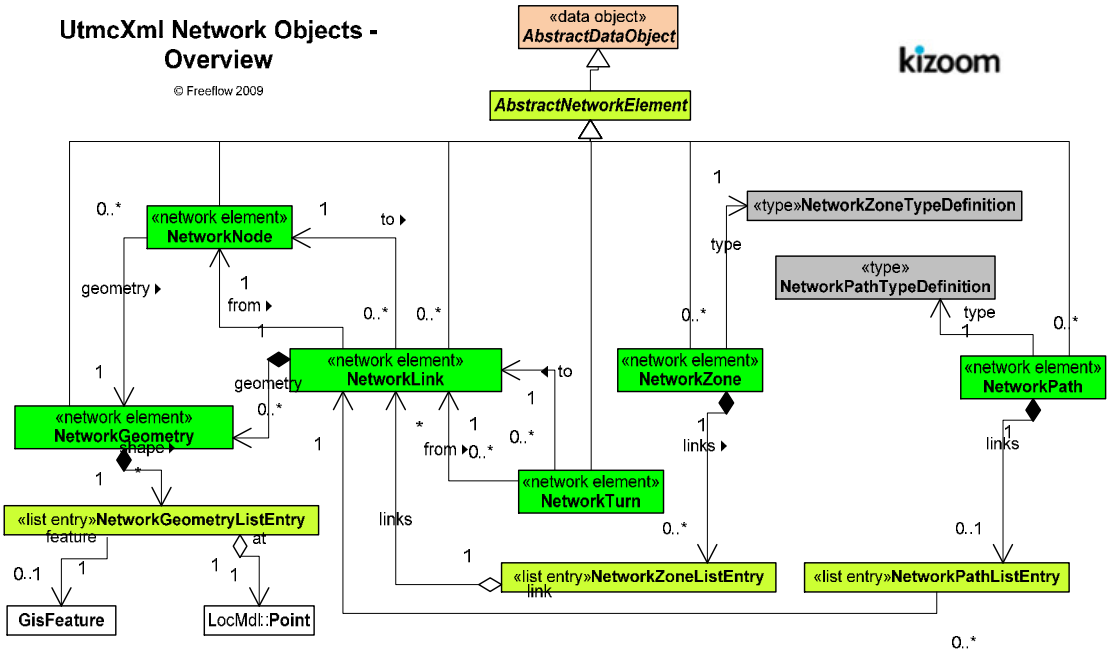


Figure 5-1 UML diagram of network objects - Overview

© Copyright Kizoom 2007-2009

UTMC NETWORK OBJECTS – DETAILS

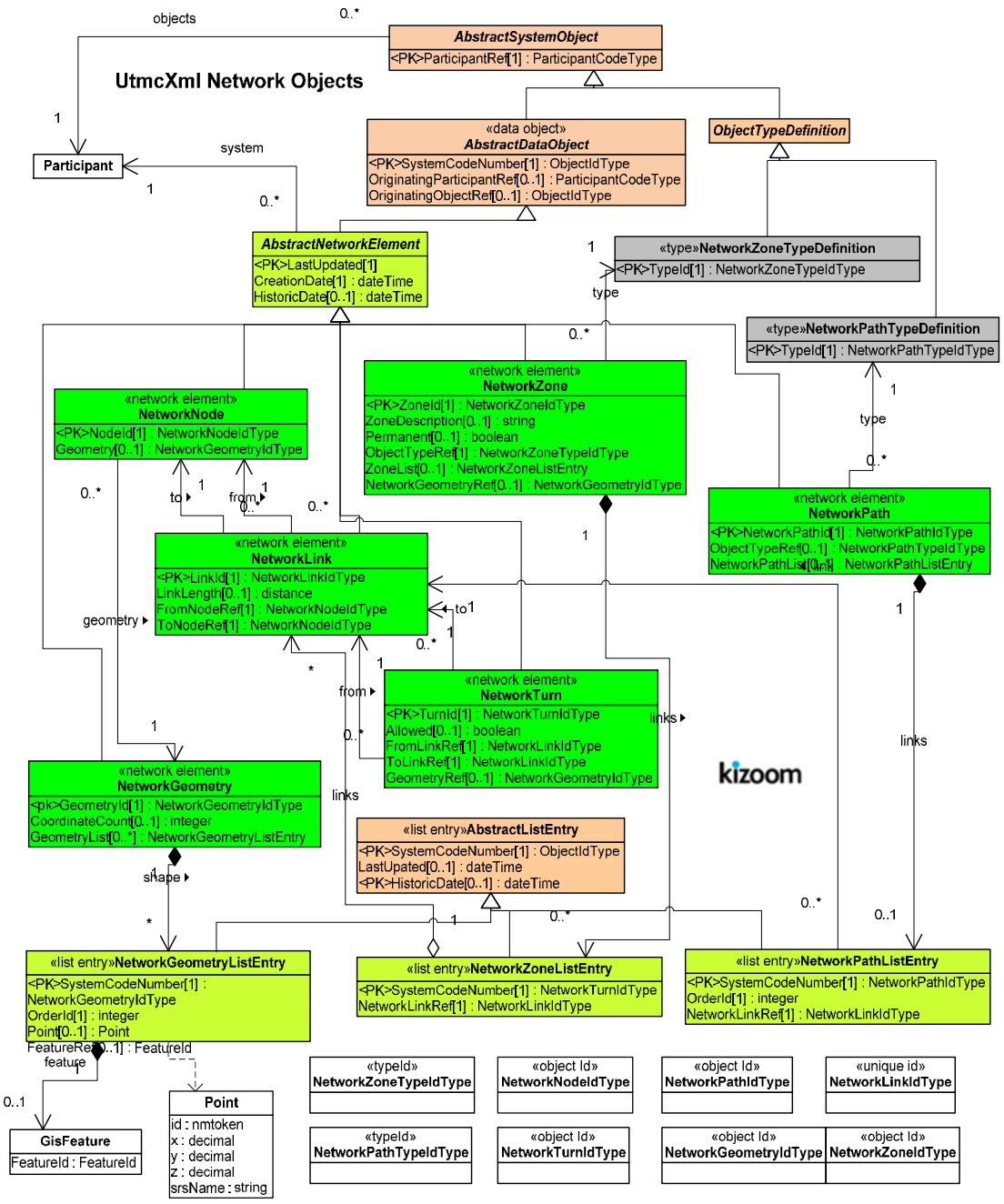


Figure 5-2 UML diagram of network objects - Details

© Copyright Kizoom 2007-2009

DEVICE OBJECTS

UTMC ACCESS CONTROL

UTMC ACCESS CONTROL – OVERVIEW

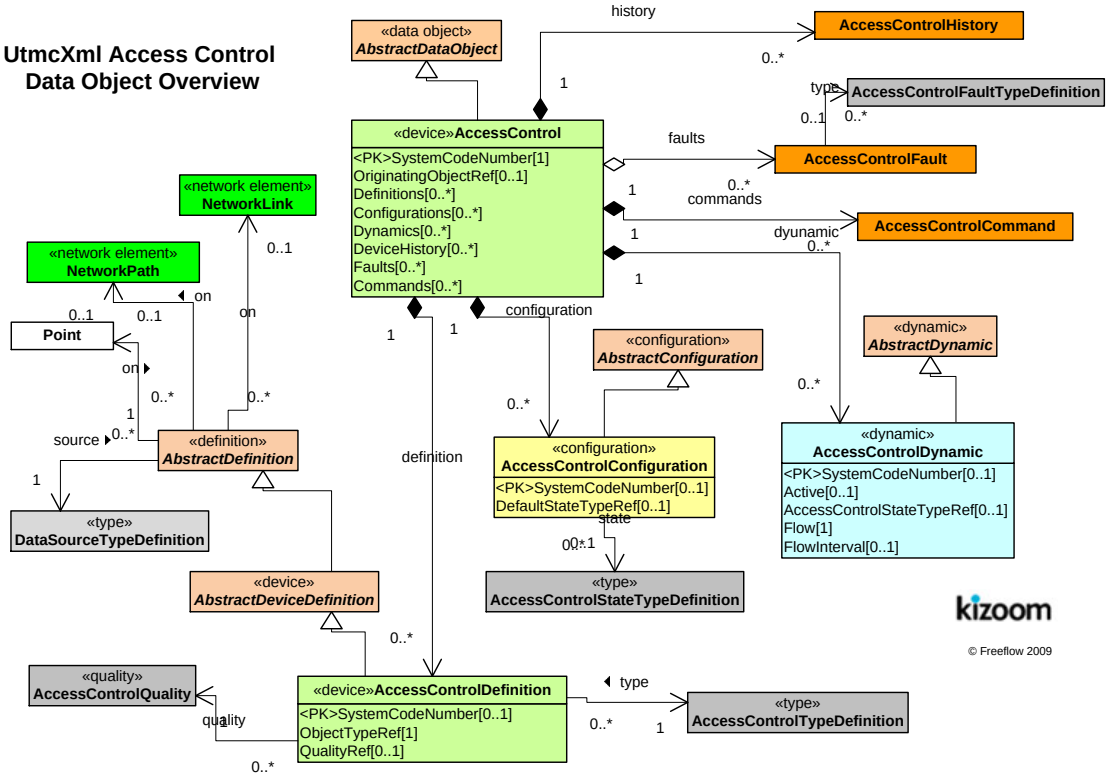


Figure 5-3 UML diagram of UTMC Access Control - Overview

UTMC ACCESS CONTROL – DETAILS

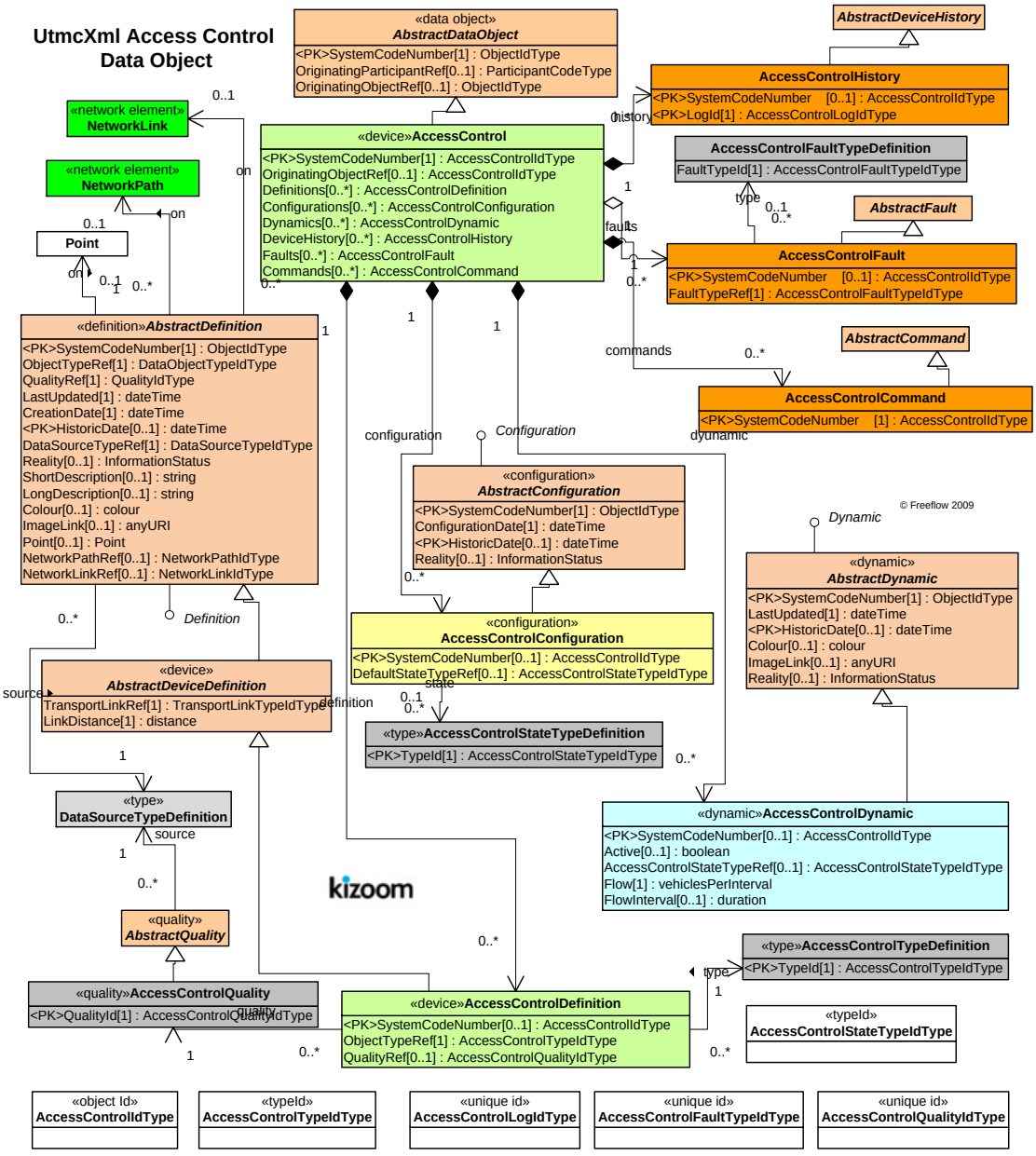


Figure 5-4 UML diagram of UTMC Access Control - Details

© Copyright Kizoom 2007-2009

UTMC AIR QUALITY

UTMC AIR QUALITY – OVERVIEW

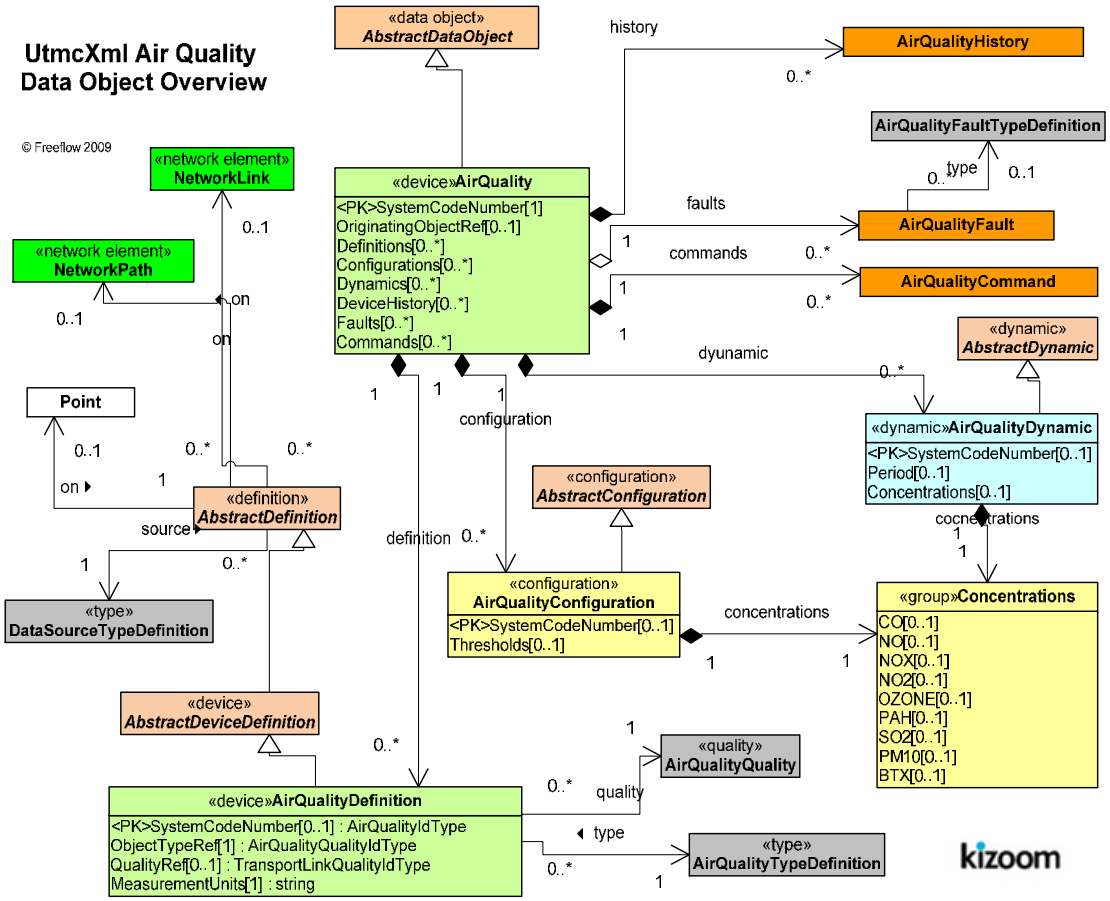


Figure 5-5 UML diagram of UTMC Air Quality- Overview

© Copyright Kizoom 2007-2009

UTMC AIR QUALITY – DETAILS

© Copyright Kizoom 2007-2009

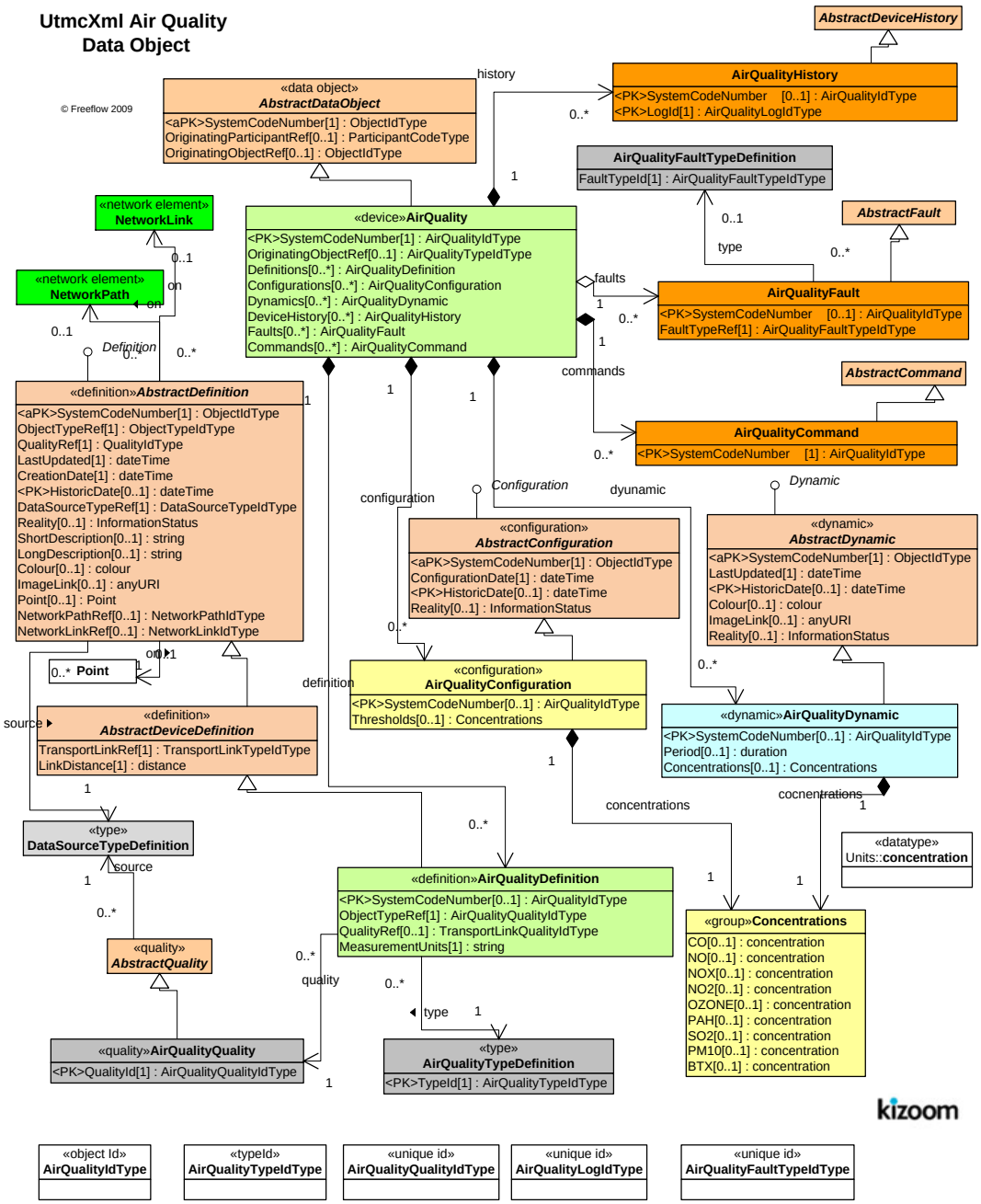


Figure 5-6 UML diagram of UTMC Air Quality- Details

UTMC CAR PARK

UTMC CAR PARK – OVERVIEW

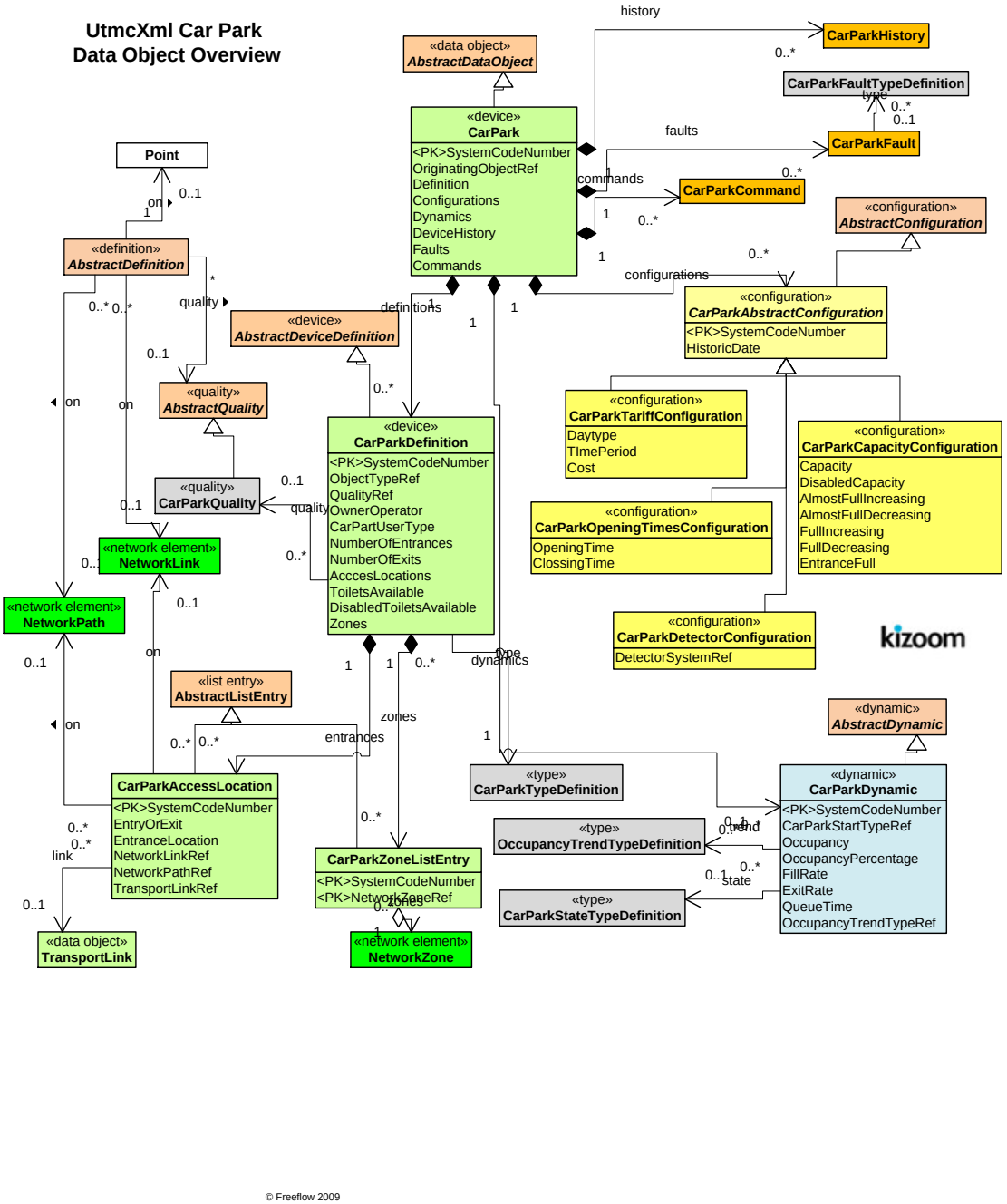


Figure 5-7 UML diagram of UTMC Car Park - Overview

UTMC CAR PARK – DETAILS

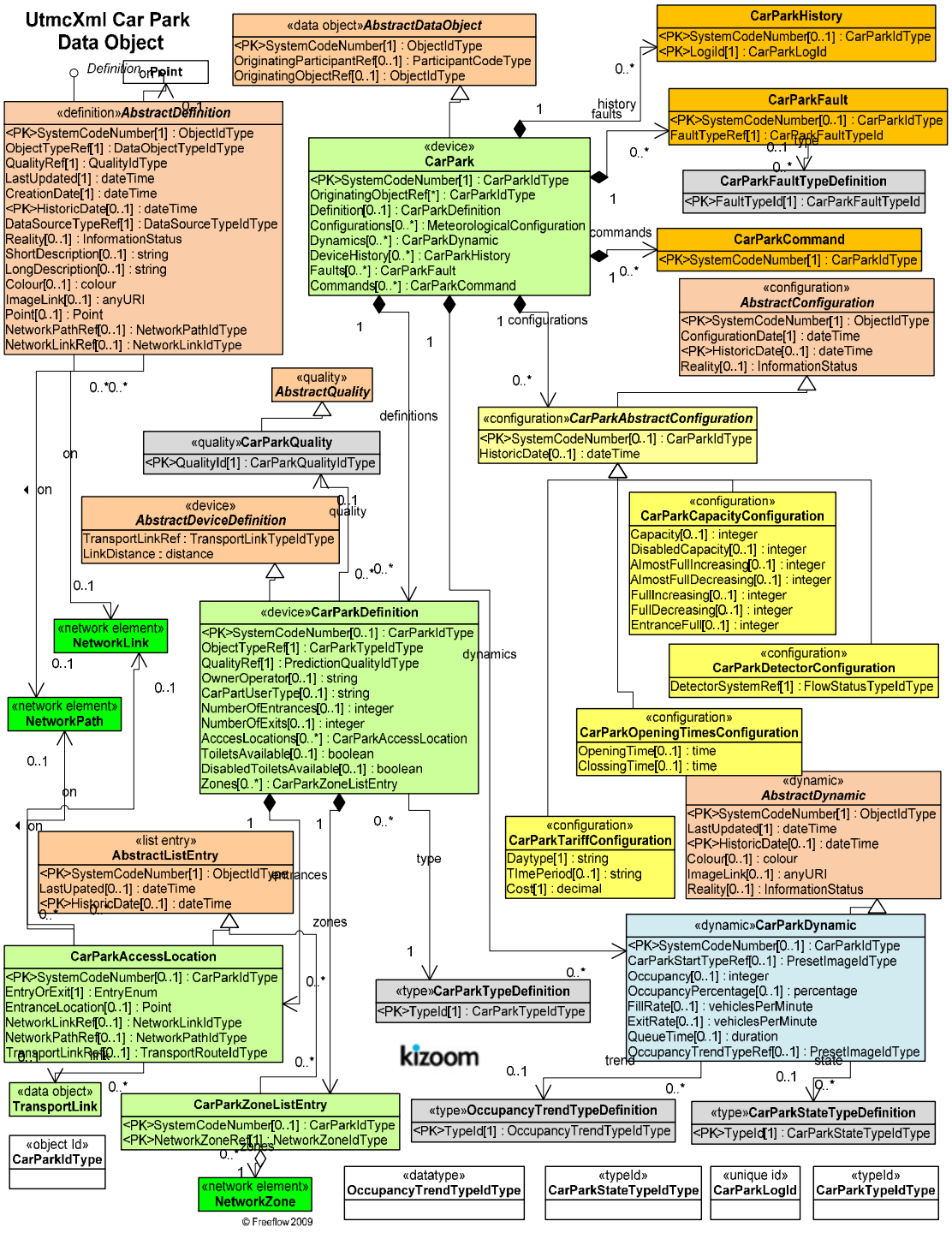


Figure 5-8 UML diagram of UTMC Car Park - Details

UTMC CCTV

UTMC CCTV- OVERVIEW

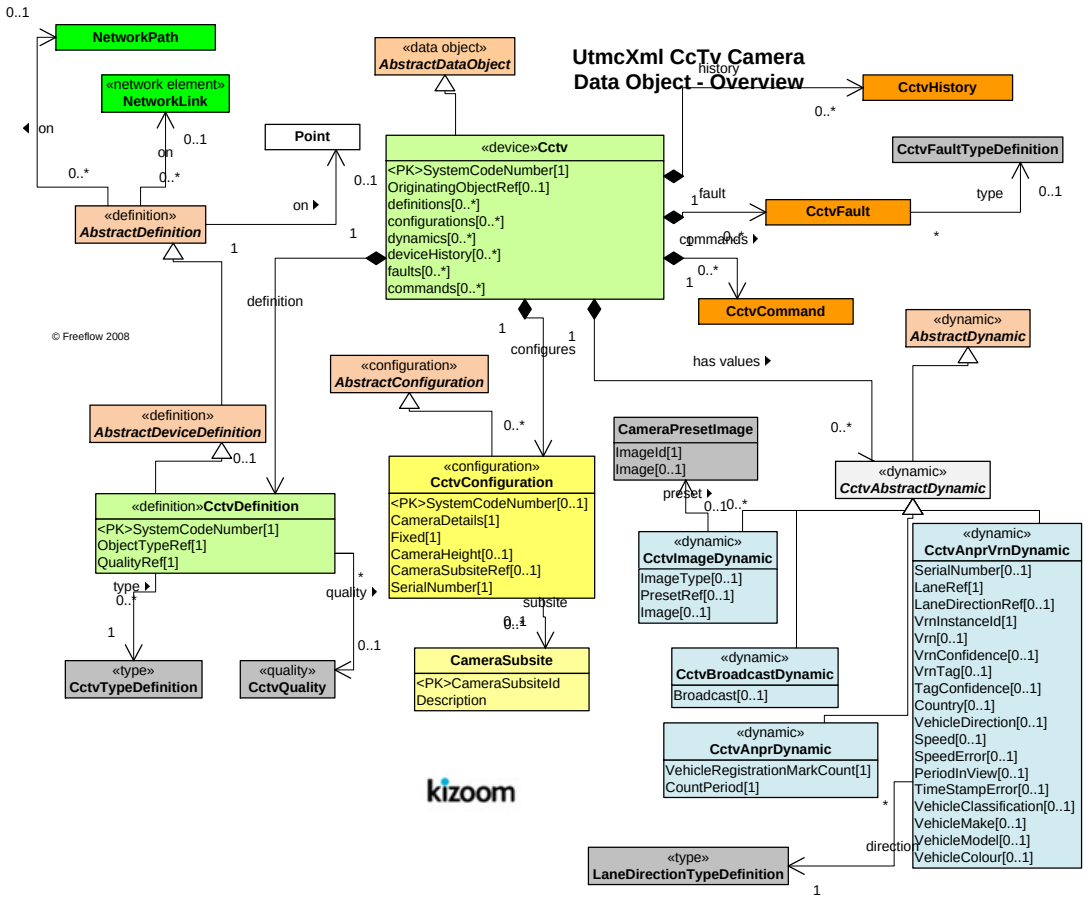


Figure 5-9 UML diagram of UTMC Cctv Overview

© Copyright Kizoom 2007-2009

UTMC CCTV – DETAILS

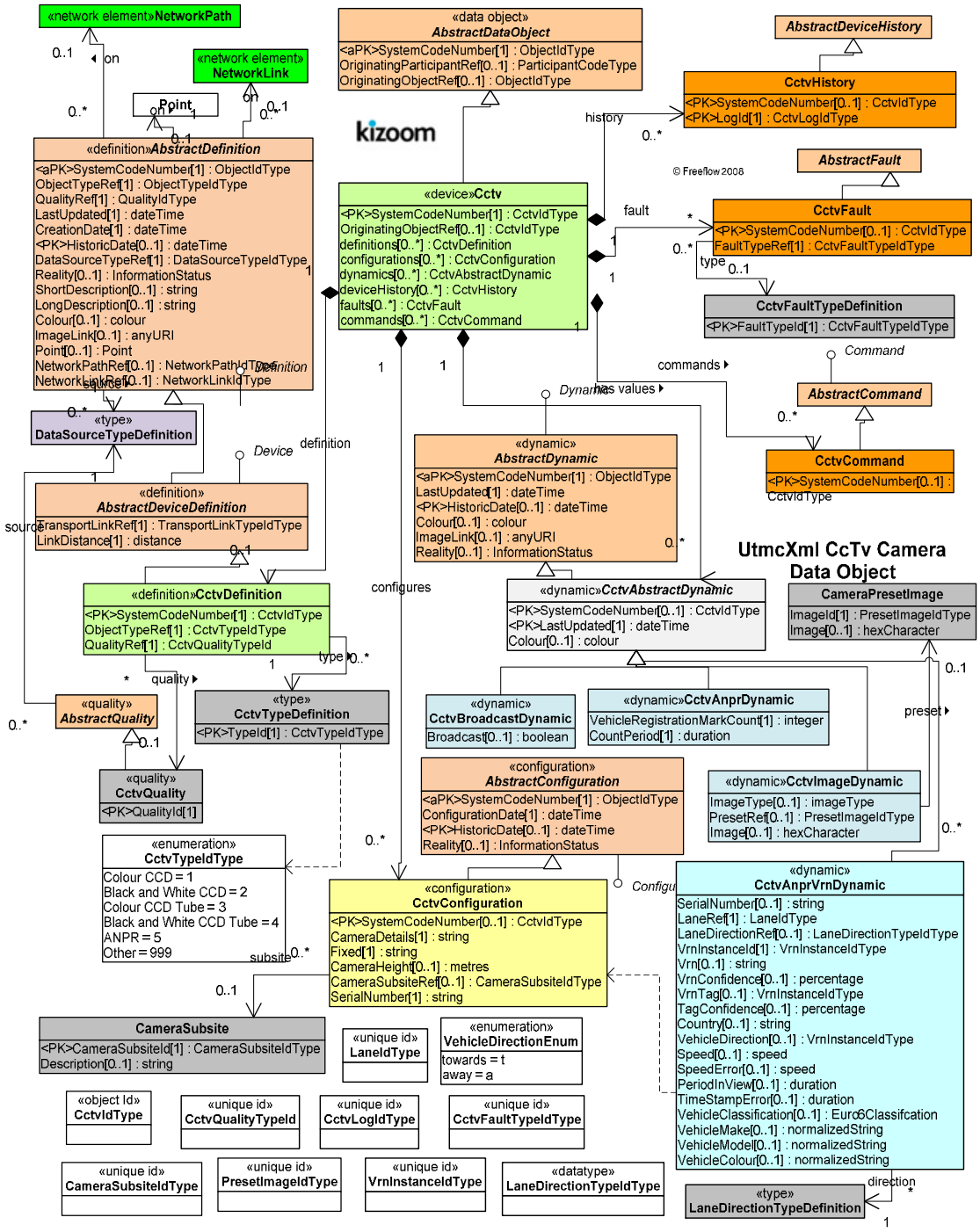


Figure 5-10 UML diagram of UTMC Cctv Details

© Copyright Kizoom 2007-2009

UTMC DETECTOR

UTMC DETECTOR – OVERVIEW

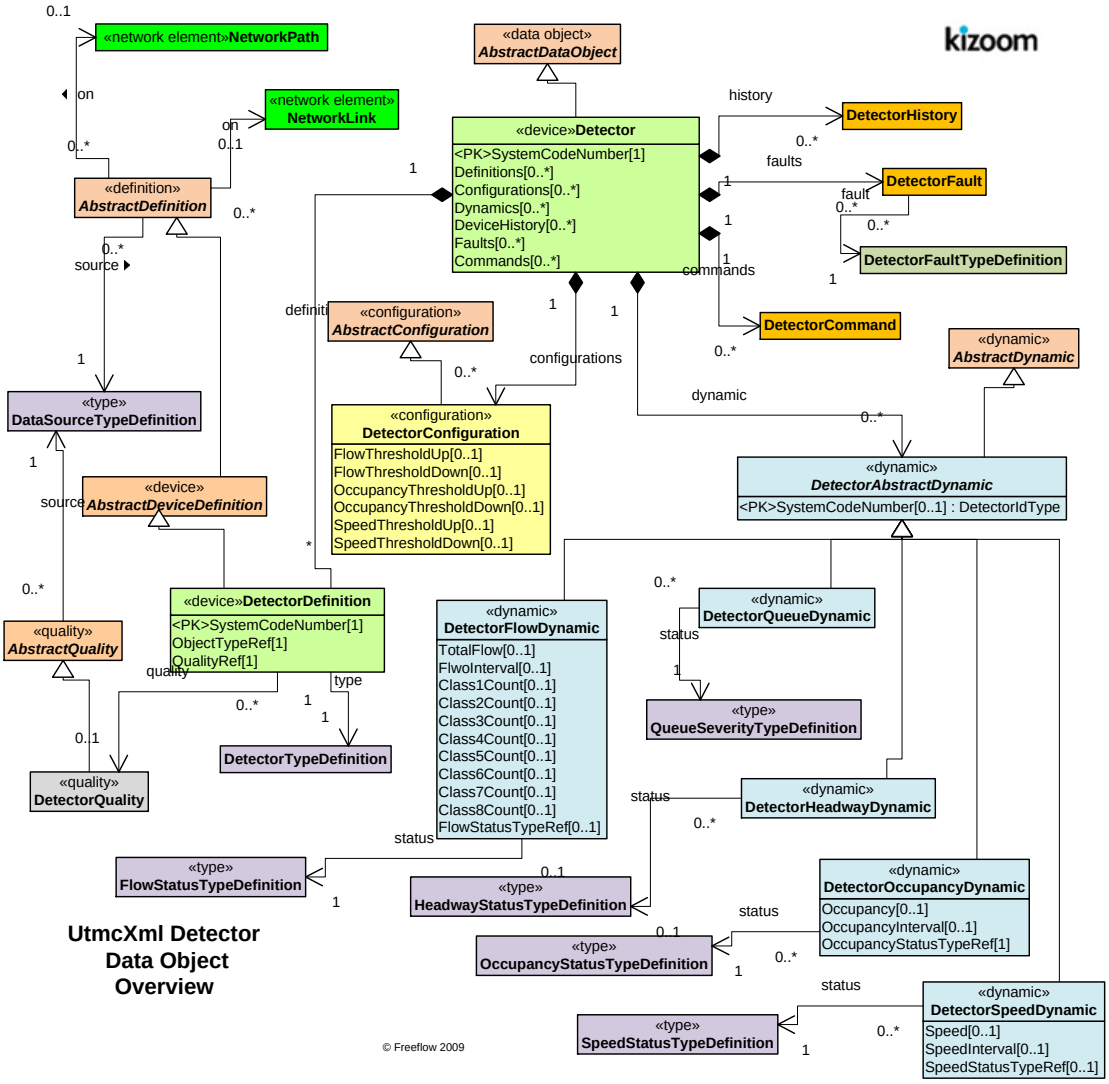


Figure 5-11 UML diagram of UTMC Detector - Overview

© Copyright Kizoom 2007-2009

UTMC DETECTOR – DETAILS

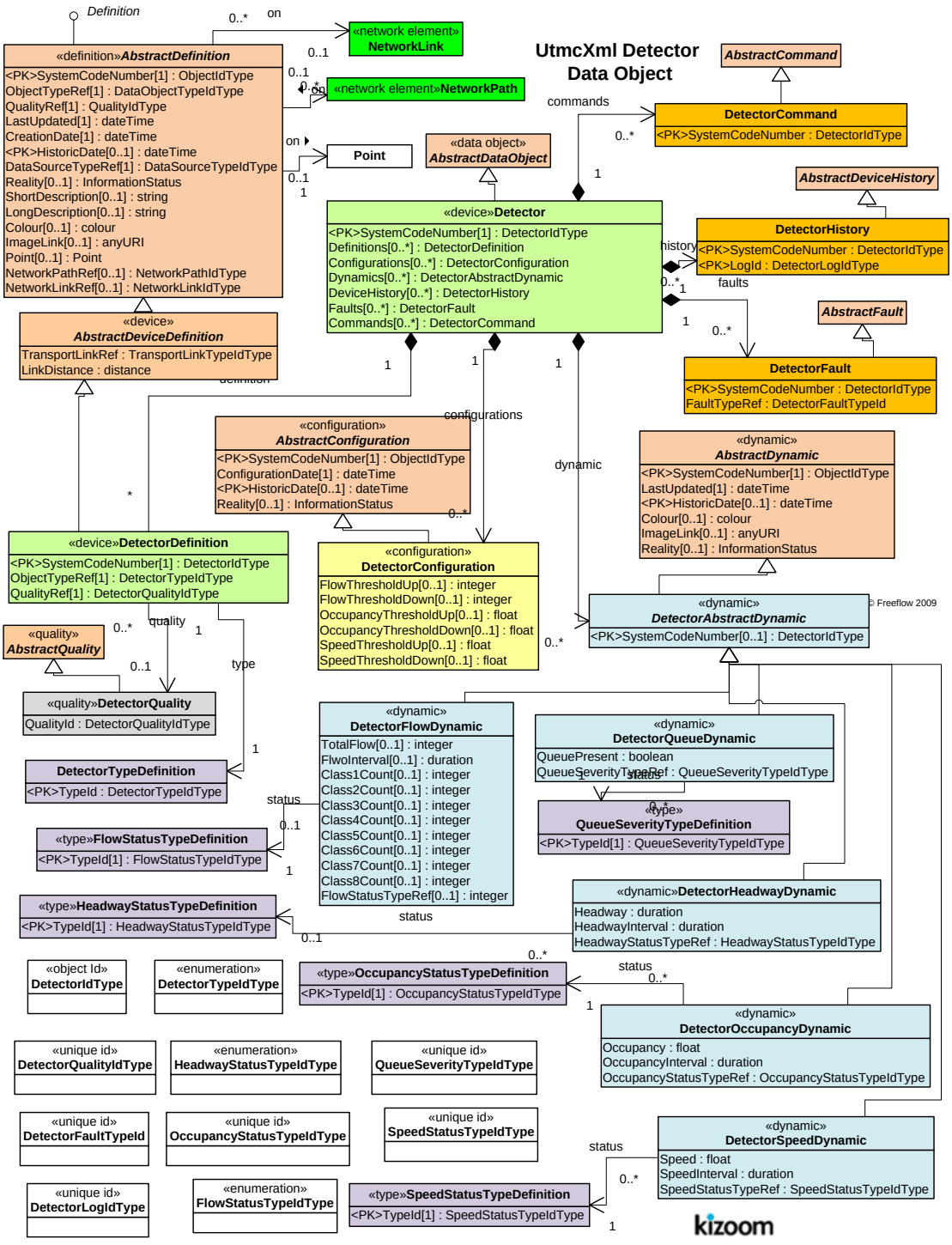


Figure 5-12 UML diagram of UTMC Detector - Details

© Copyright Kizoom 2007-2009

UTMC METEOROLOGICAL

UTMC METEOROLOGICAL – OVERVIEW

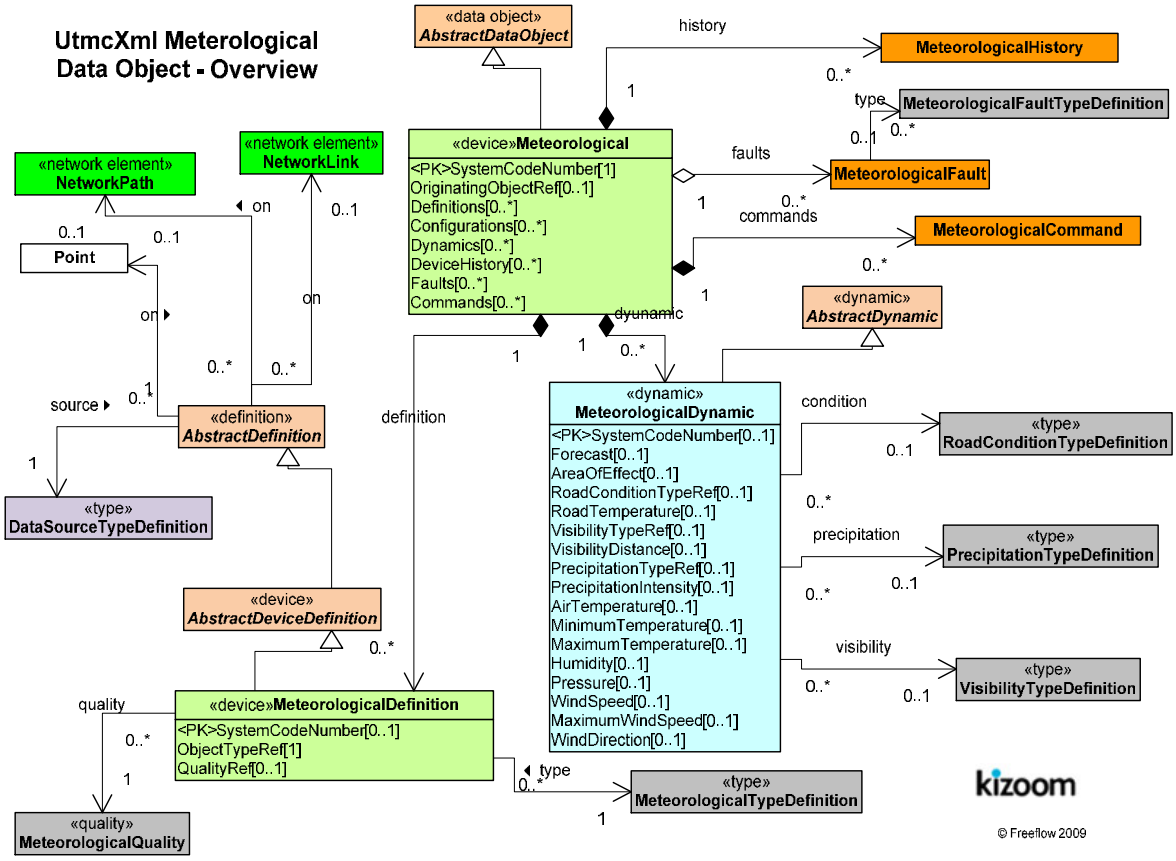


Figure 5-13 UML diagram of UTMC Meteorological - Overview

© Copyright Kizoom 2007-2009

UTMC METEOROLOGICAL - DETAILS

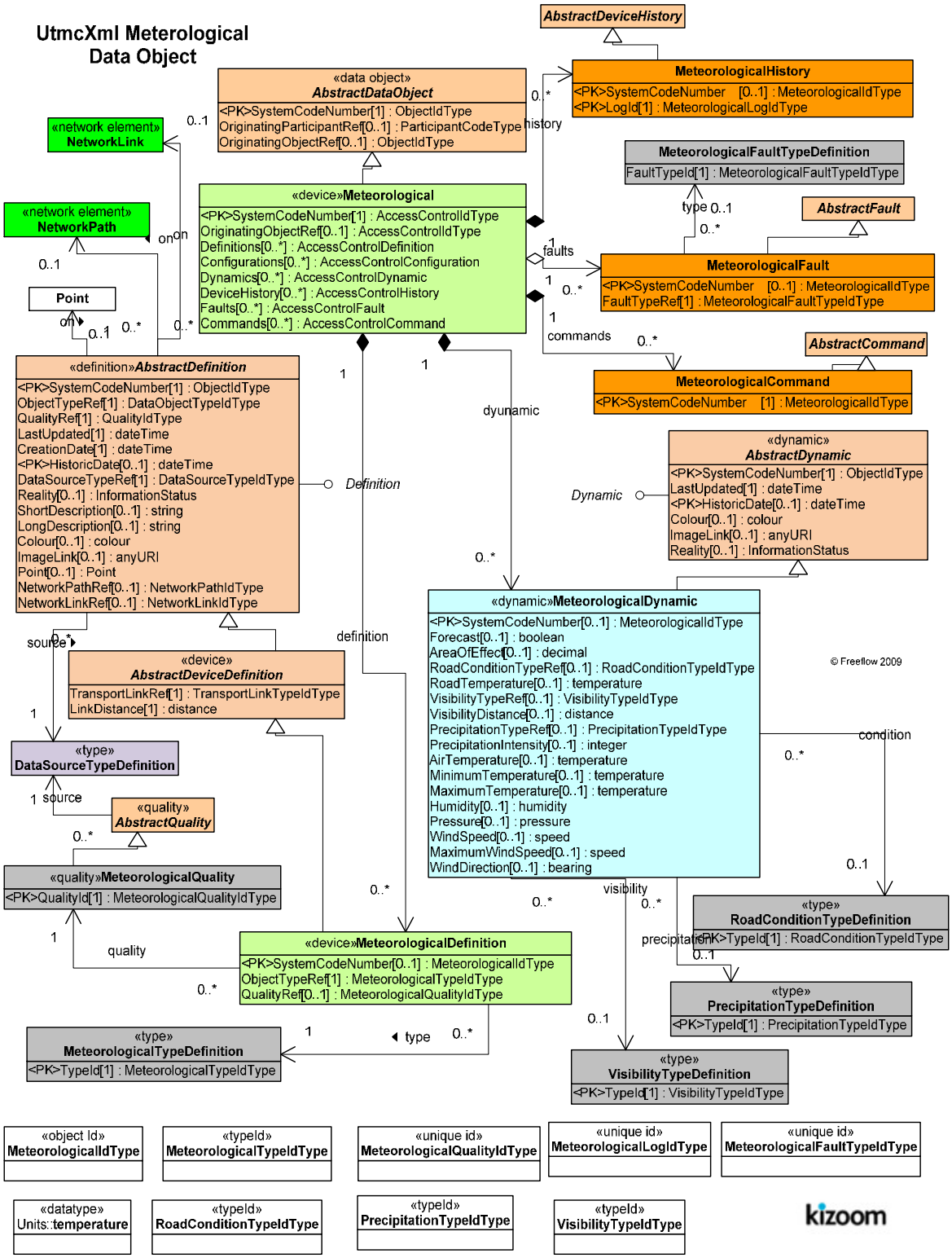


Figure 5-14 UML diagram of UTMC Meteorological - Details

© Copyright Kizoom 2007-2009

UTMC TRAFFIC SIGNAL

UTMC TRAFFIC SIGNAL – OVERVIEW

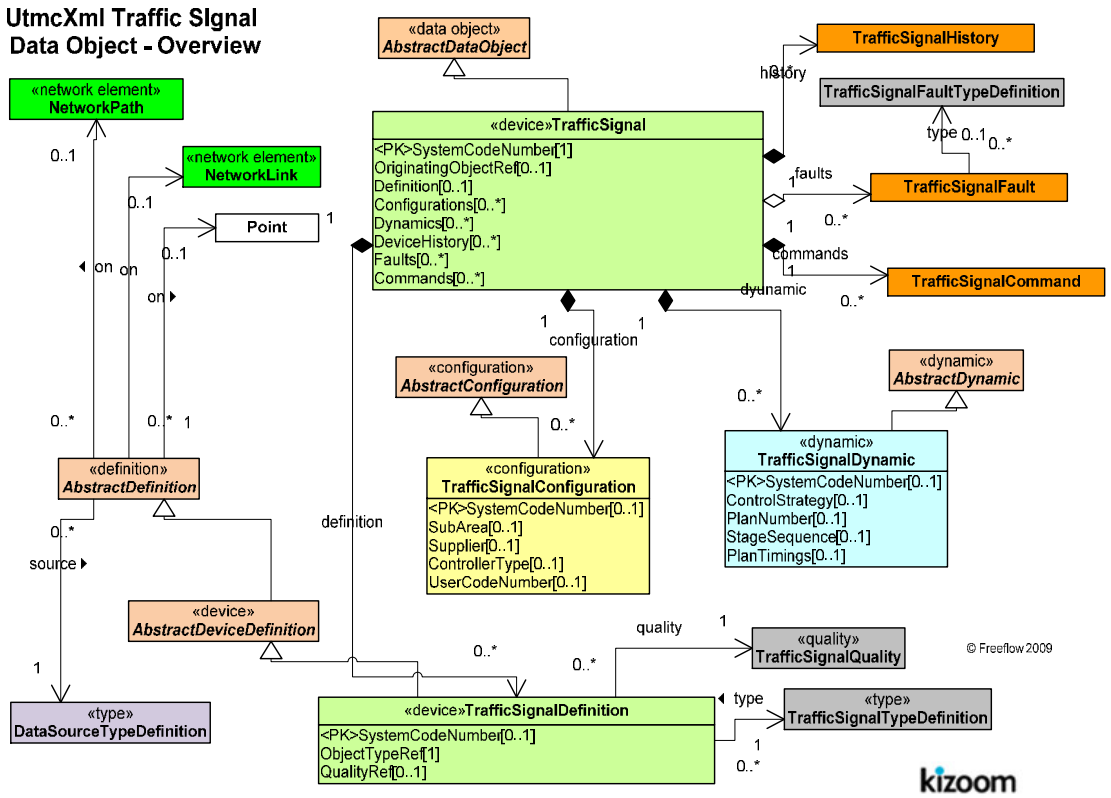


Figure 5-15 UML diagram of UTMC Traffic Signal - Overview

© Copyright Kizoom 2007-2009

UTMC TRAFFIC SIGNAL - DETAILS

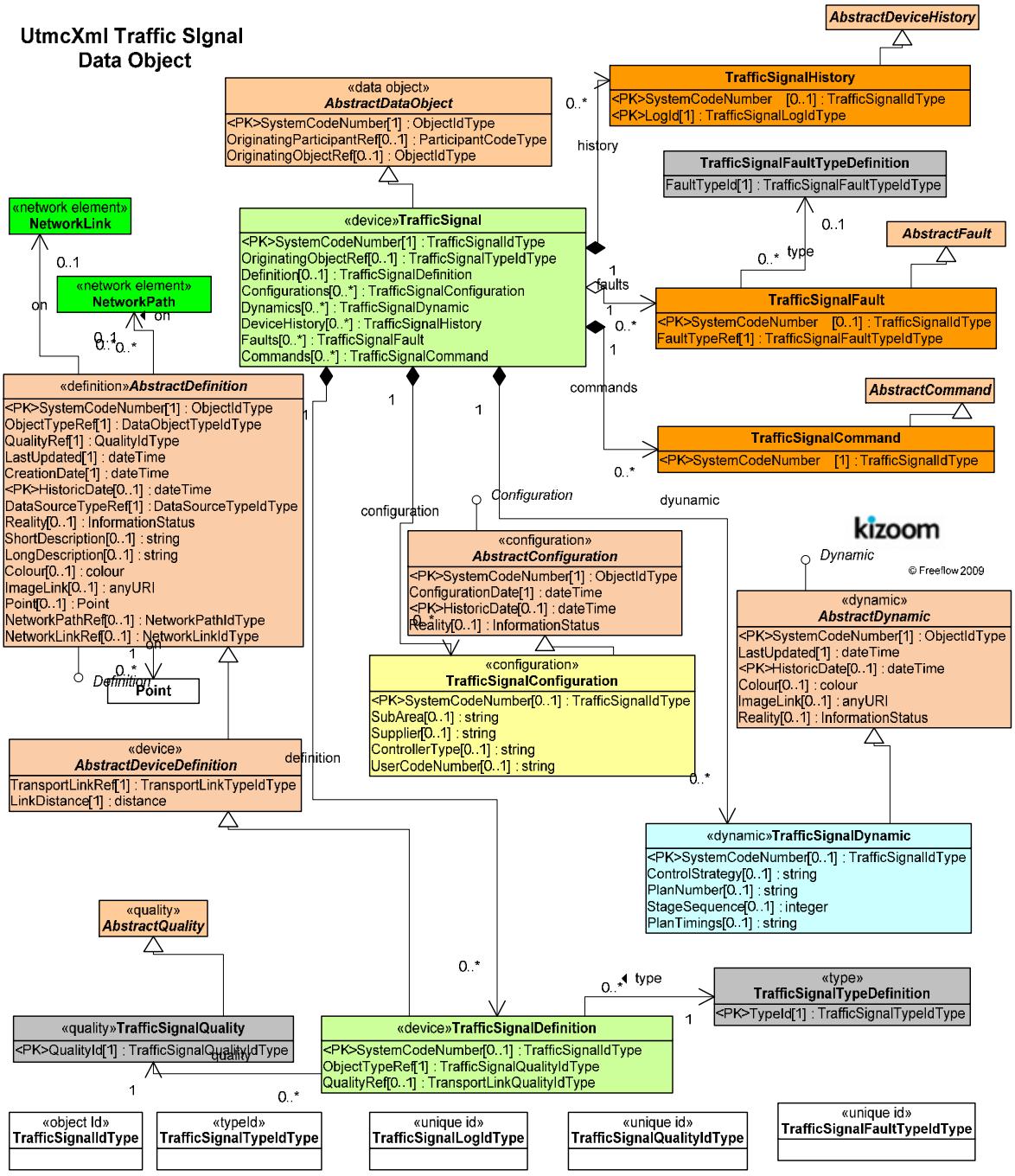


Figure 5-16 UML diagram of UTMC Traffic Signal - Details

© Copyright Kizoom 2007-2009

UTMC VARIABLE MESSAGE SIGN

UTMC VARIABLE MESSAGE SIGN – OVERVIEW

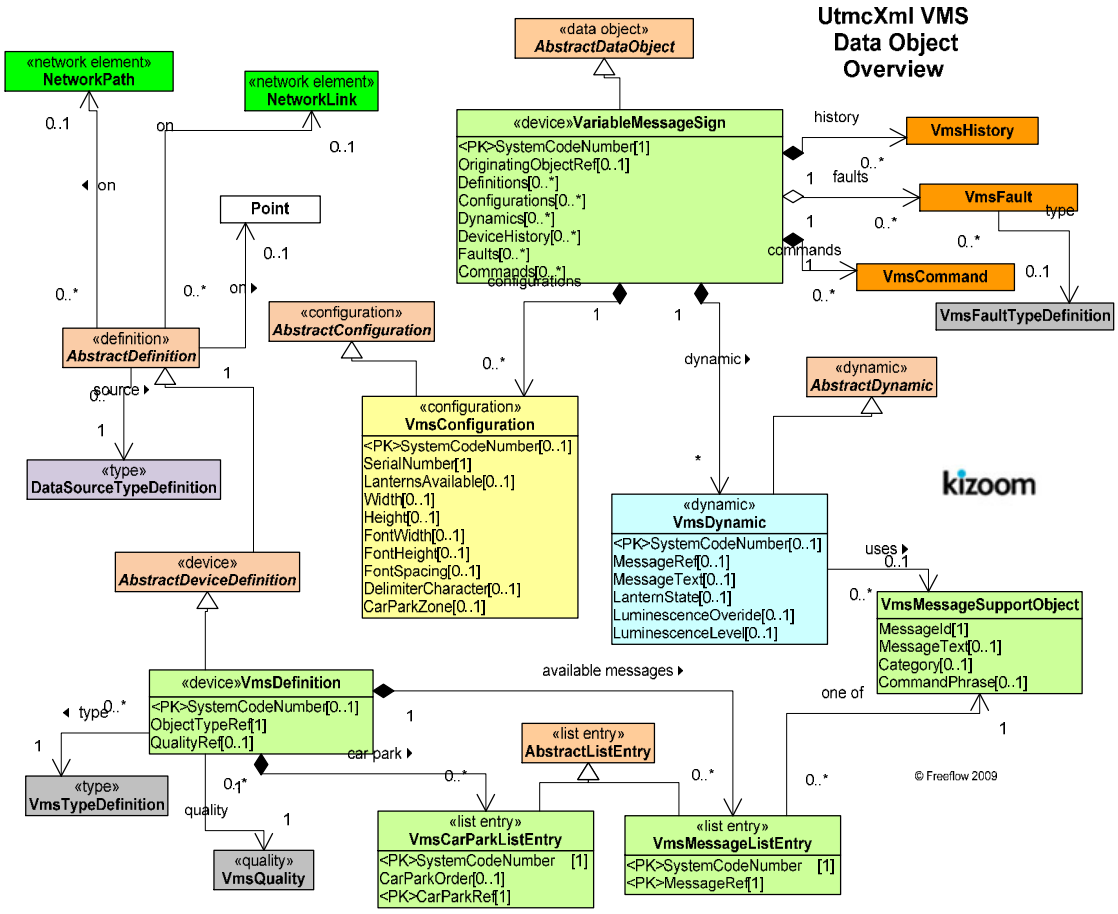


Figure 5-17 UML diagram of UTMC VMS - Overview

© Copyright Kizoom 2007-2009

UTMC VARIABLE MESSAGE SIGN - DETAILS

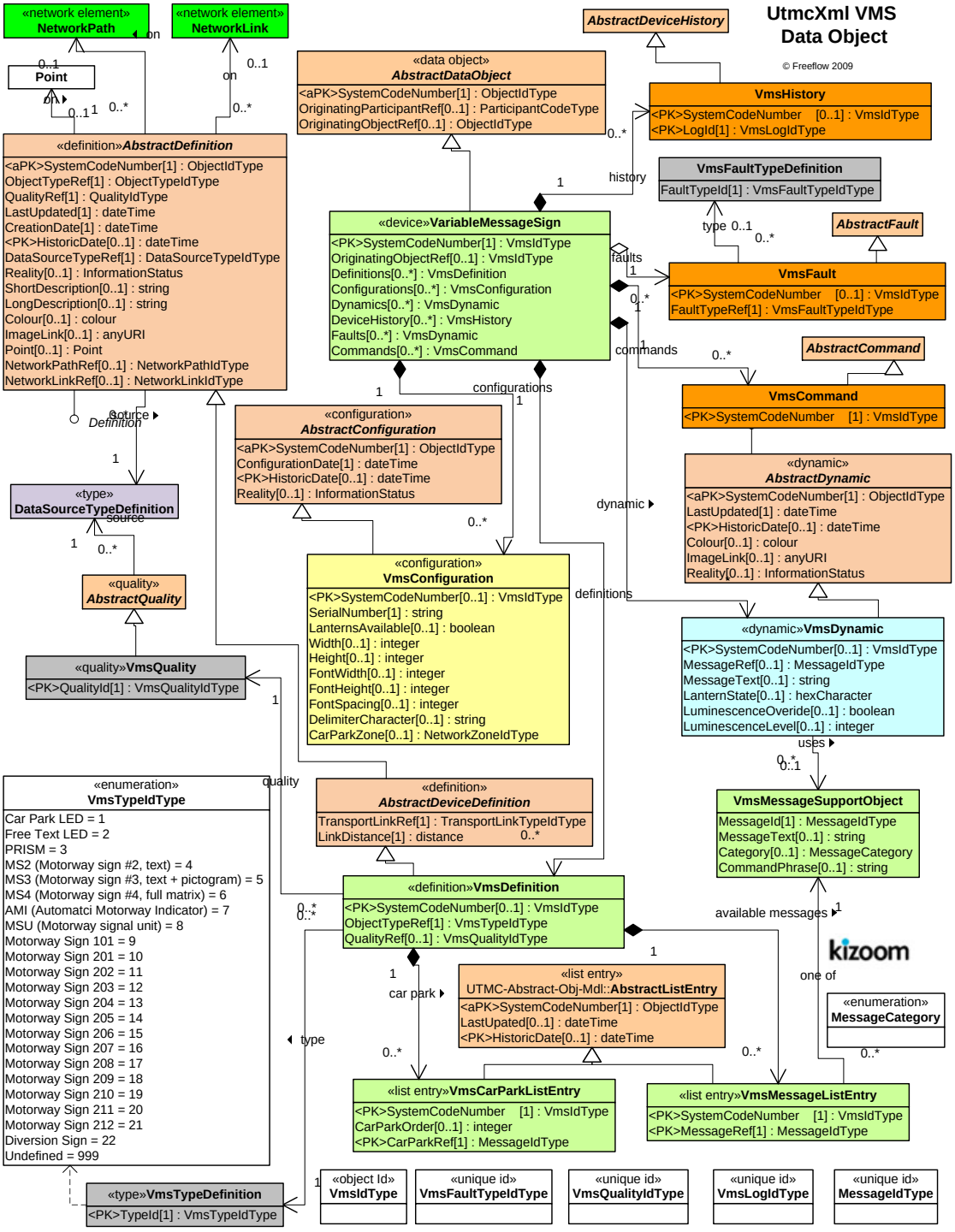


Figure 5-18 UML diagram of UTMC VMS - Details

© Copyright Kizoom 2007-2009

TRANSPORT LINKS & TRANSPORT ROUTES

The following diagram shows the relationship between Transport Links, Routes and Network Links.

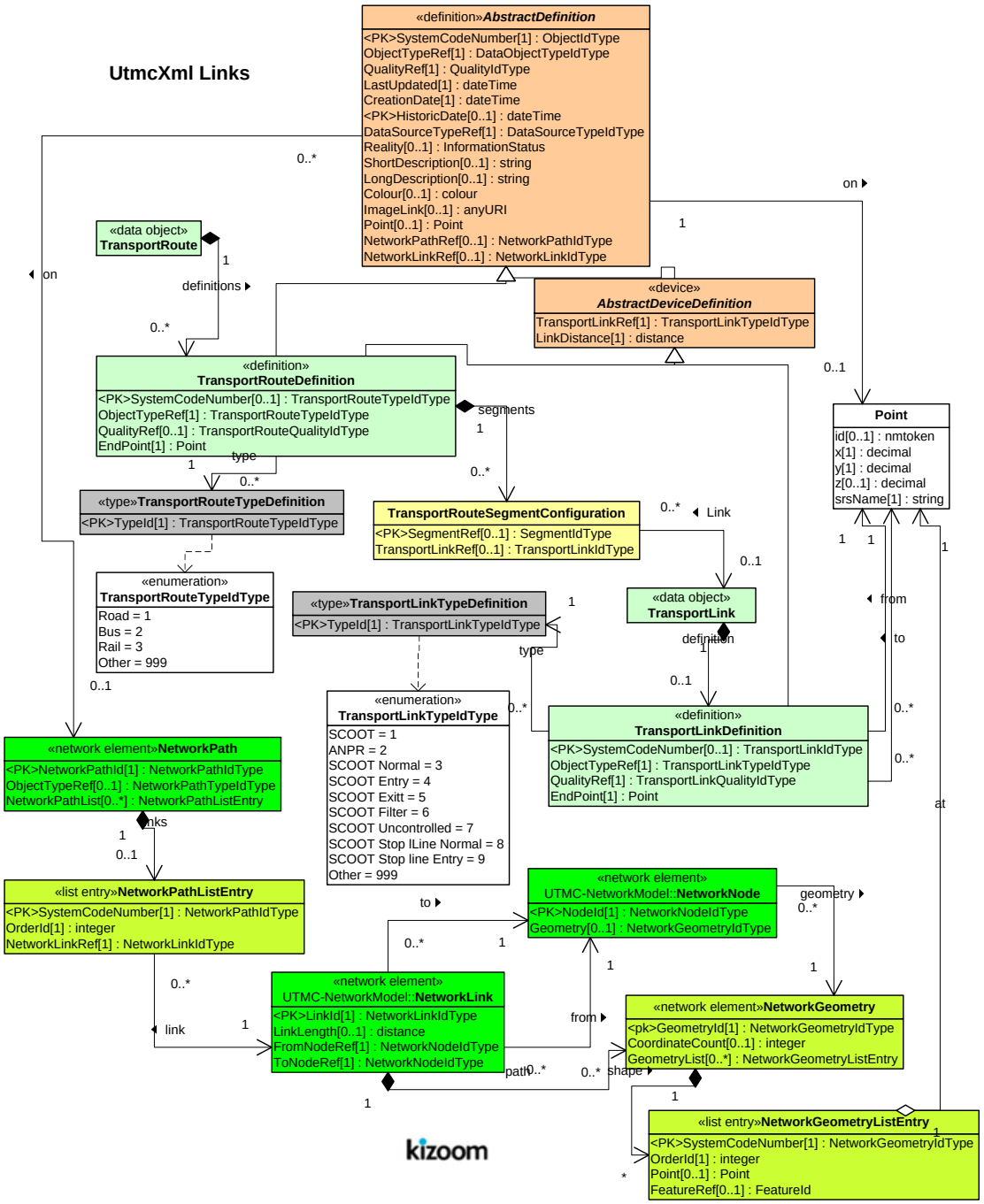


Figure 5-19 UML Diagram of Transport links and network links

© Copyright Kizoom 2007-2009

UTMC TRANSPORT LINK

UTMC TRANSPORT LINK – OVERVIEW

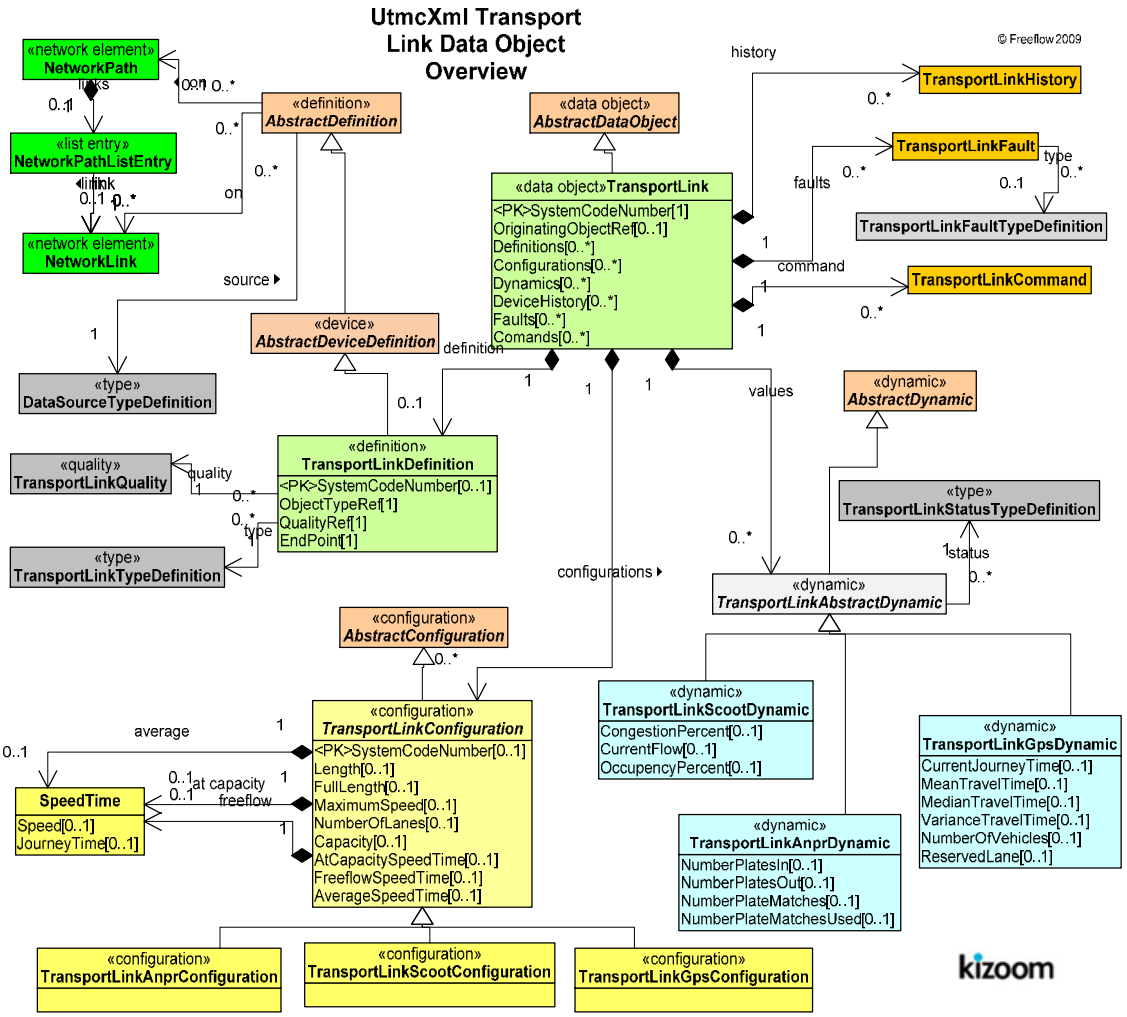


Figure 5-20 UML diagram of UTMC Transport Link - Overview

© Copyright Kizoom 2007-2009

UTMC TRANSPORT LINK - DETAILS

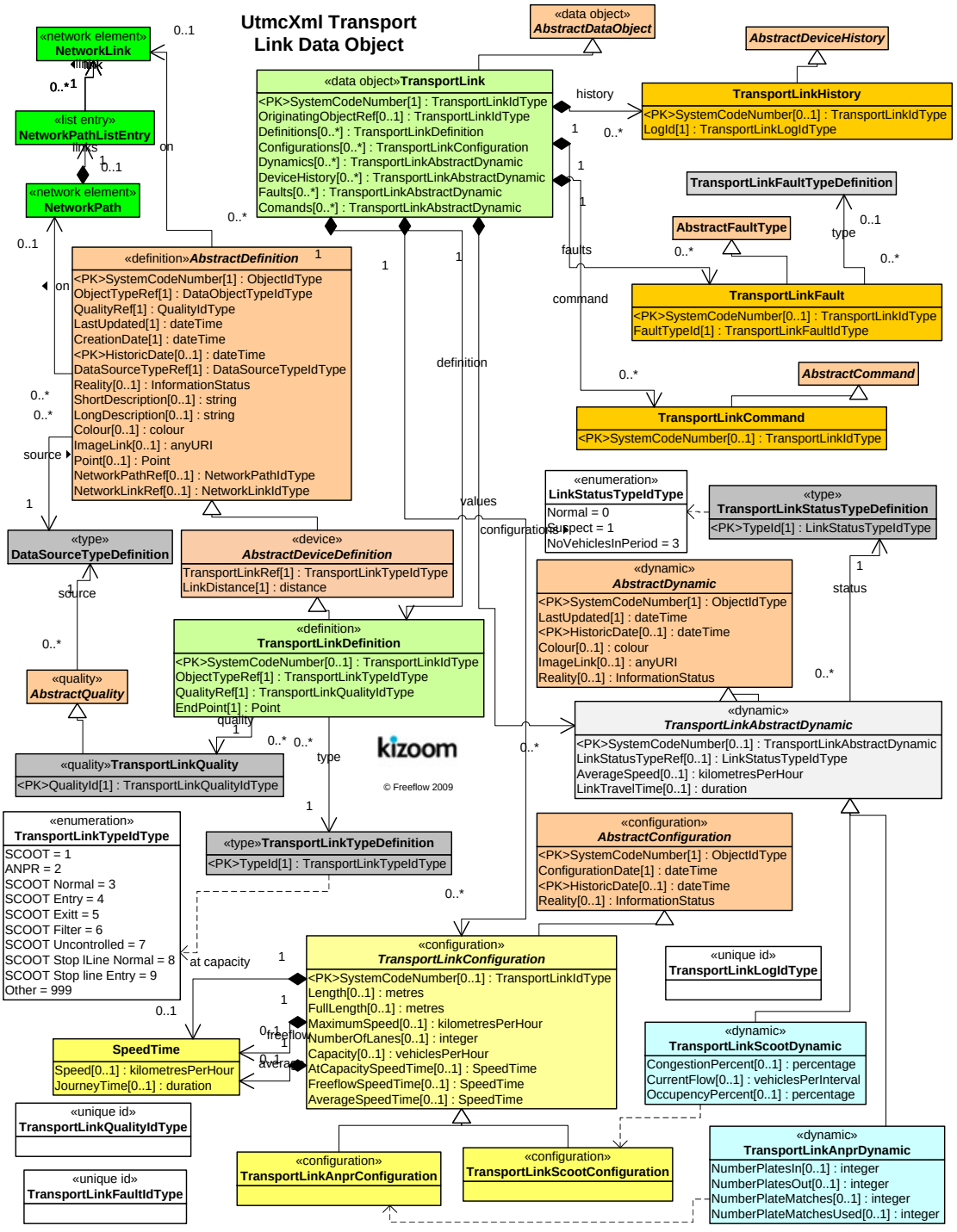


Figure 5-21 UML diagram of UTMC Transport Link - Details

© Copyright Kizoom 2007-2009

UTMC TRANSPORT ROUTE

UTMC TRANSPORT ROUTE - OVERVIEW

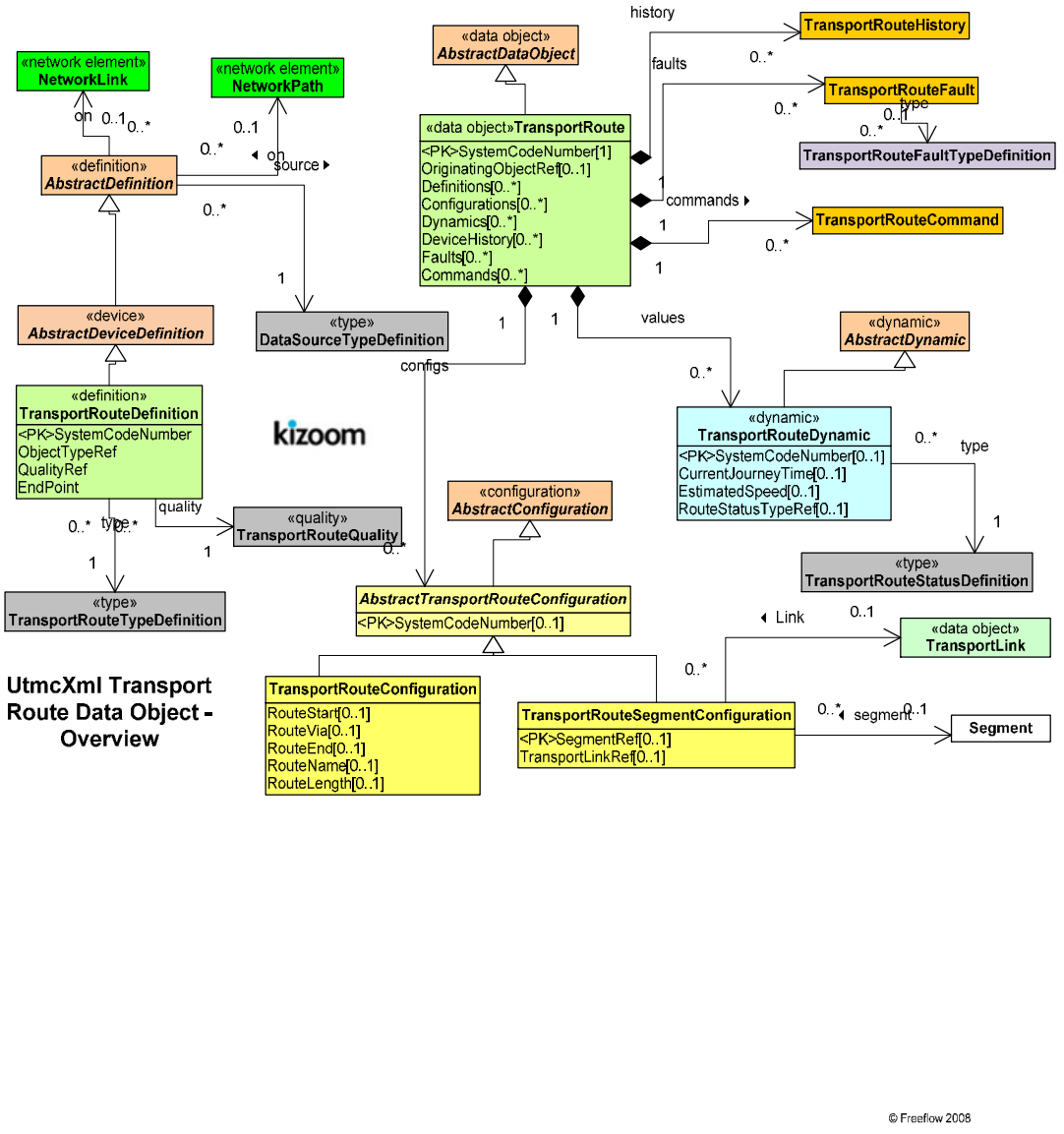


Figure 5-22 UML diagram of UTMC Transport Route - Overview

© Copyright Kizoom 2007-2009

UTMC TRANSPORT ROUTE DETAILS

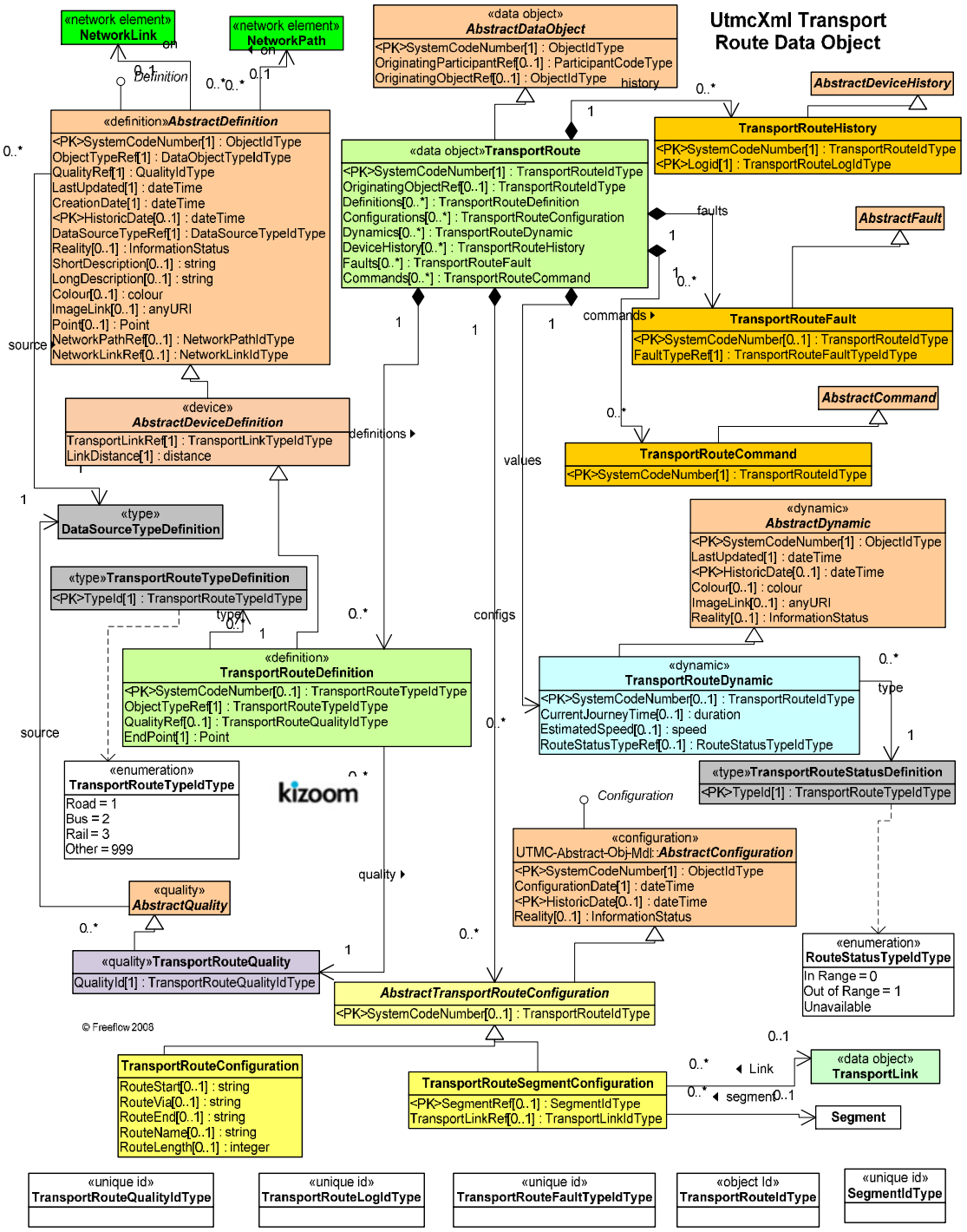


Figure 5-23 UML diagram of UTMC Transport Route - Details

© Copyright Kizoom 2007-2009

TRAFFIC EVENT OBJECTS

OVERVIEW

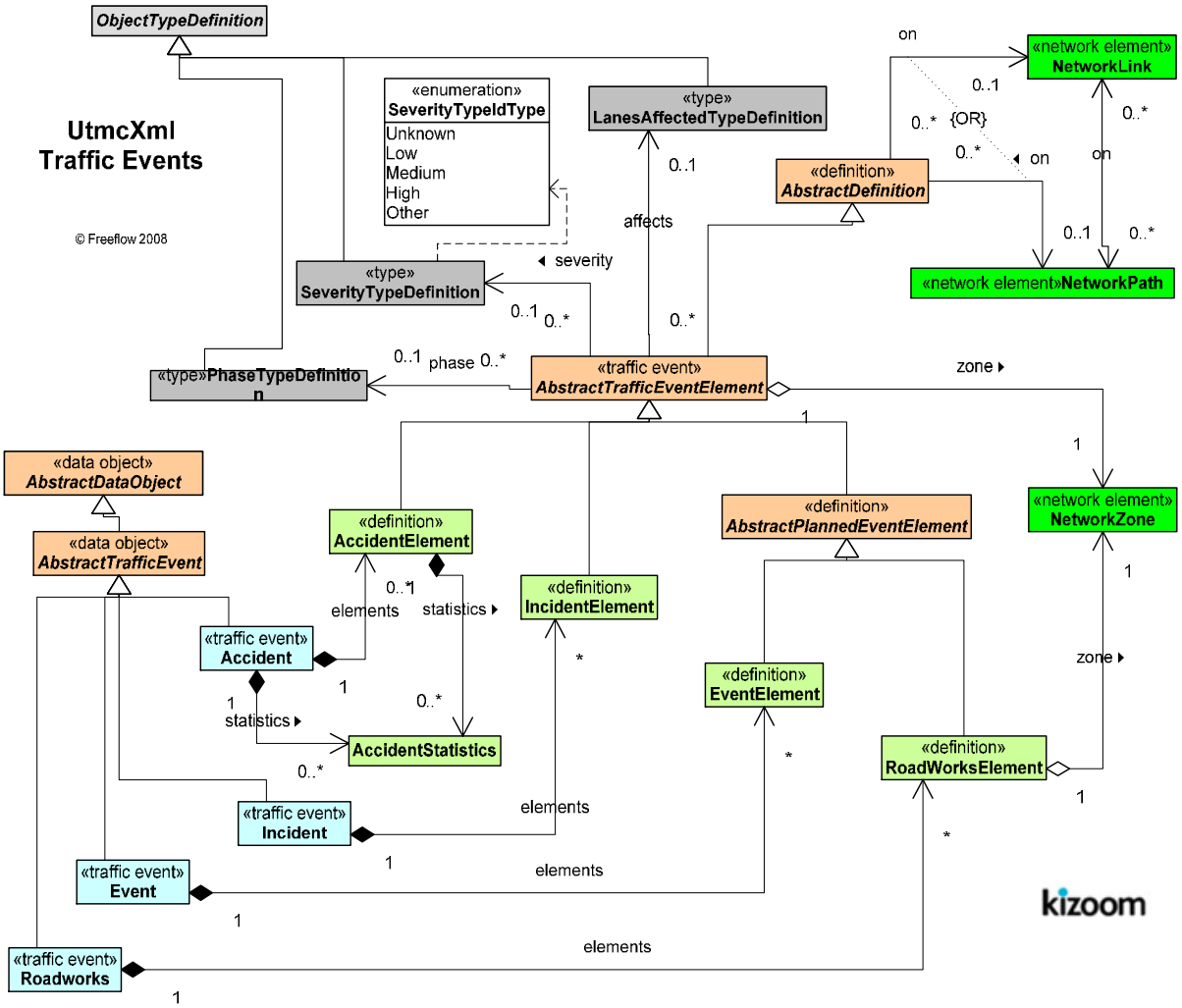
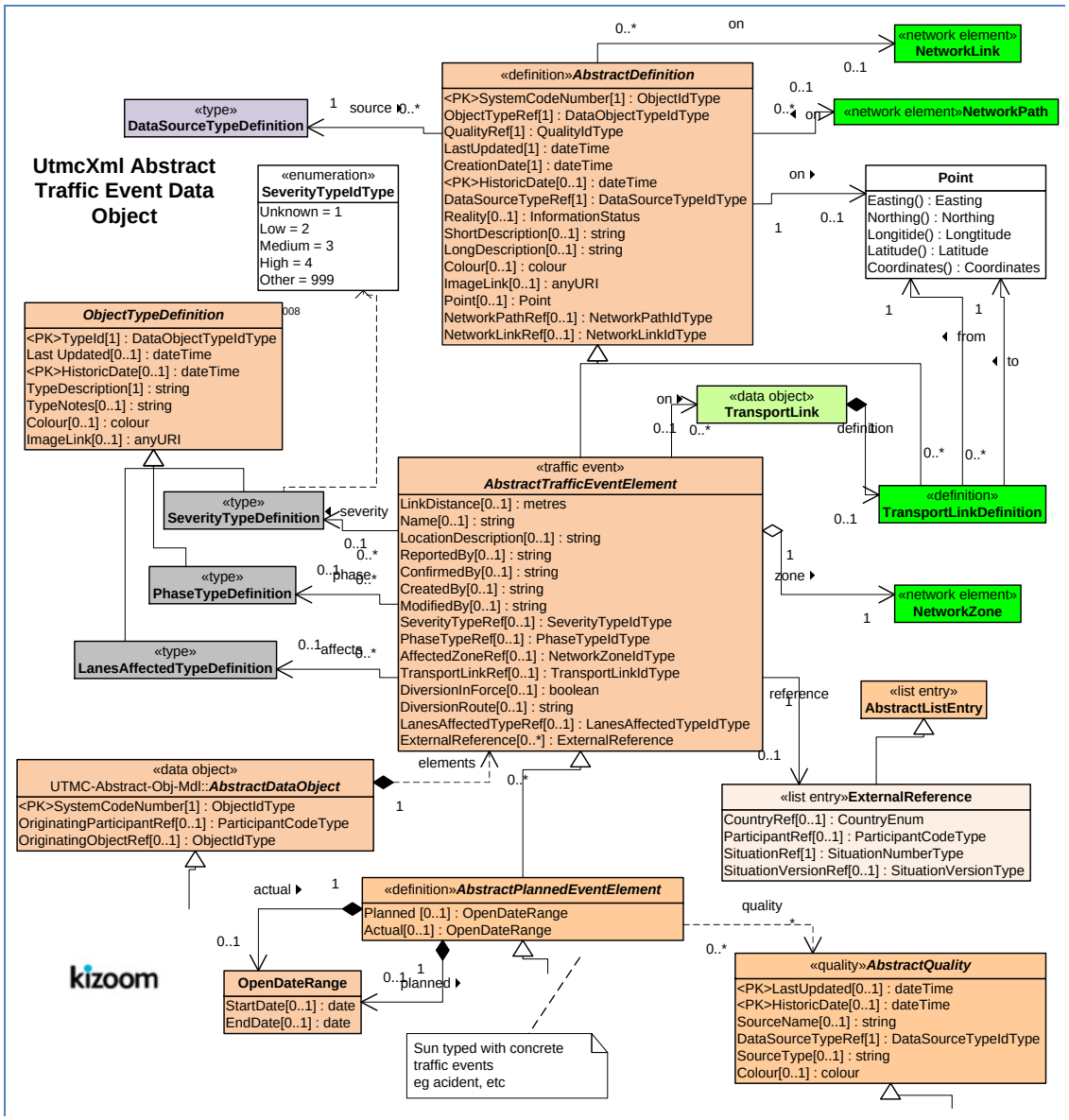


Figure 5-24 UML diagram of UTMC Abstract Traffic Event

© Copyright Kizoom 2007-2009

ABSTRACT TRAFFIC EVENT OBJECT



© Copyright Kizoom 2007-2009

UTMC ACCIDENT

UTMC ACCIDENT – OVERVIEW

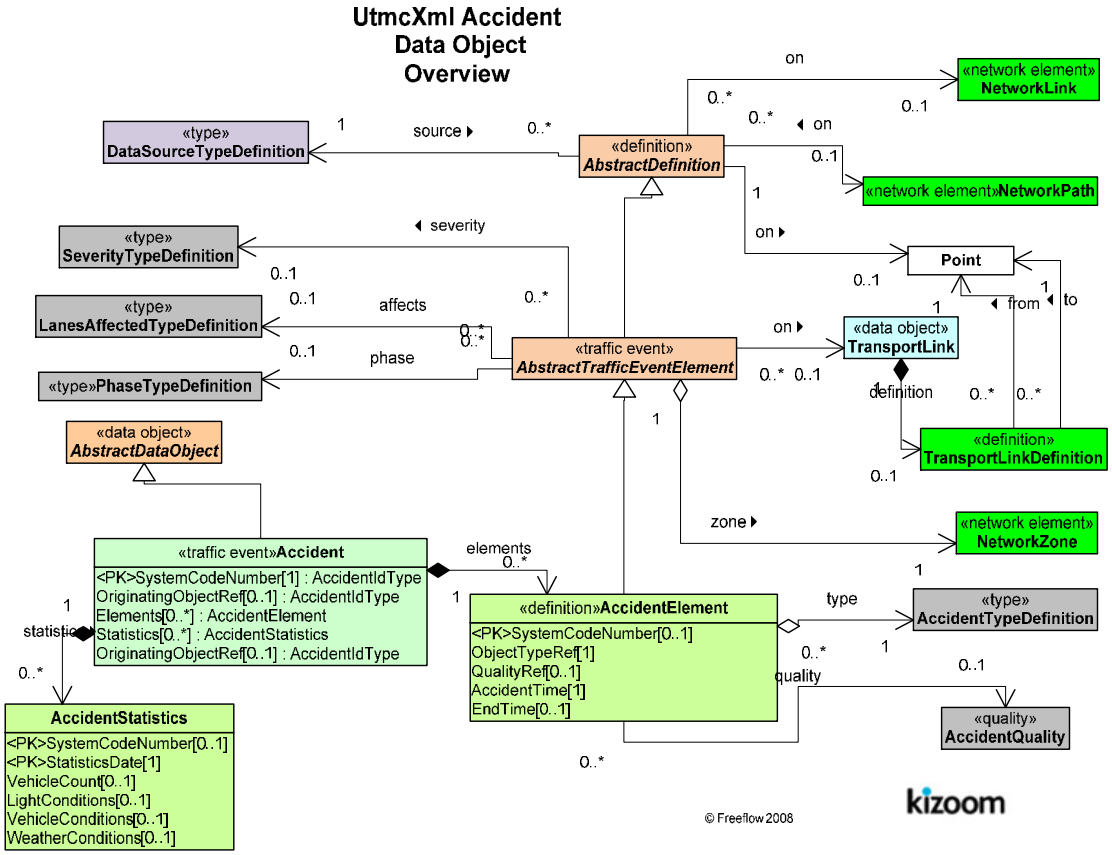


Figure 5-25 UML diagram of UTMC Accident - Overview

© Copyright Kizoom 2007-2009

UTMC ACCIDENT – DETAILS

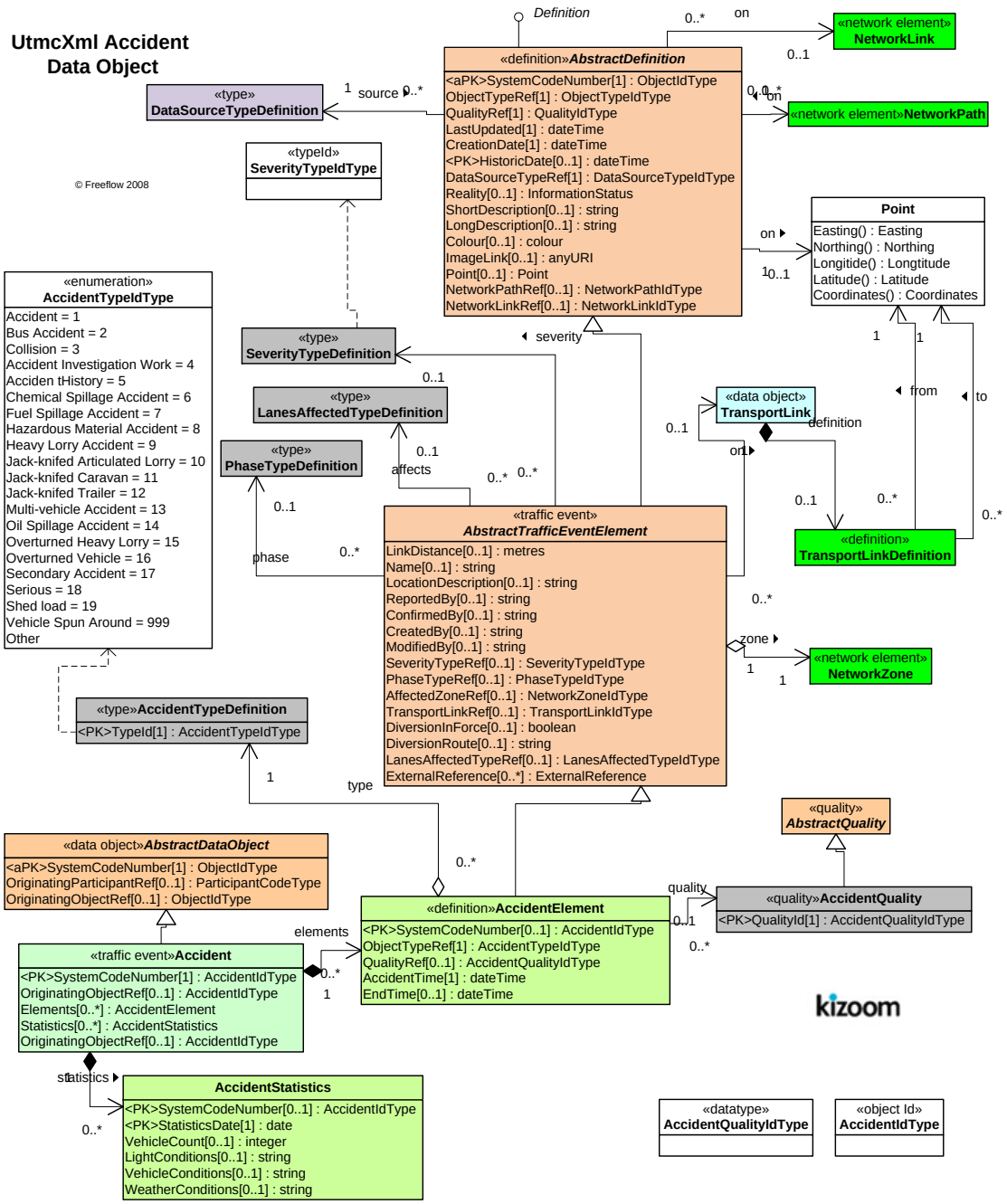


Figure 5-26 UML diagram of UTMC Accident - Details

© Copyright Kizoom 2007-2009

UTMC INCIDENT

UTMC INCIDENT – OVERVIEW

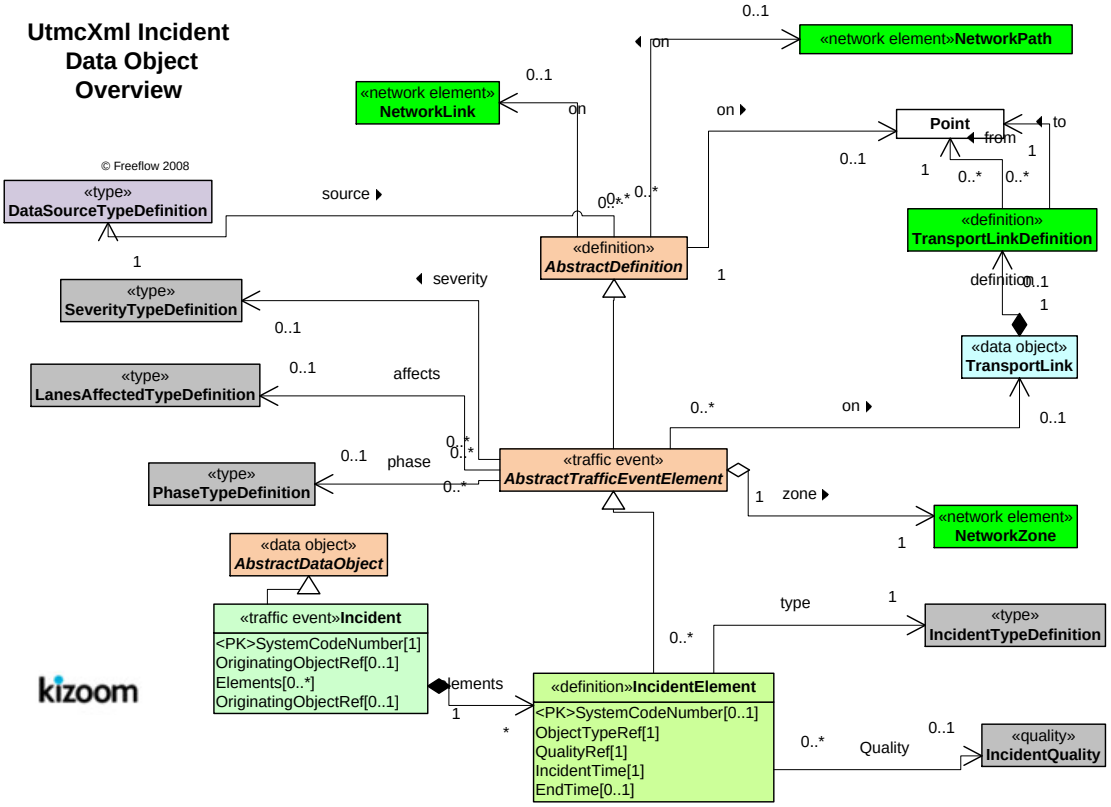


Figure 5-27 UML diagram of UTMC Incident - Overview

© Copyright Kizoom 2007-2009

UTMC INCIDENT – DETAILS

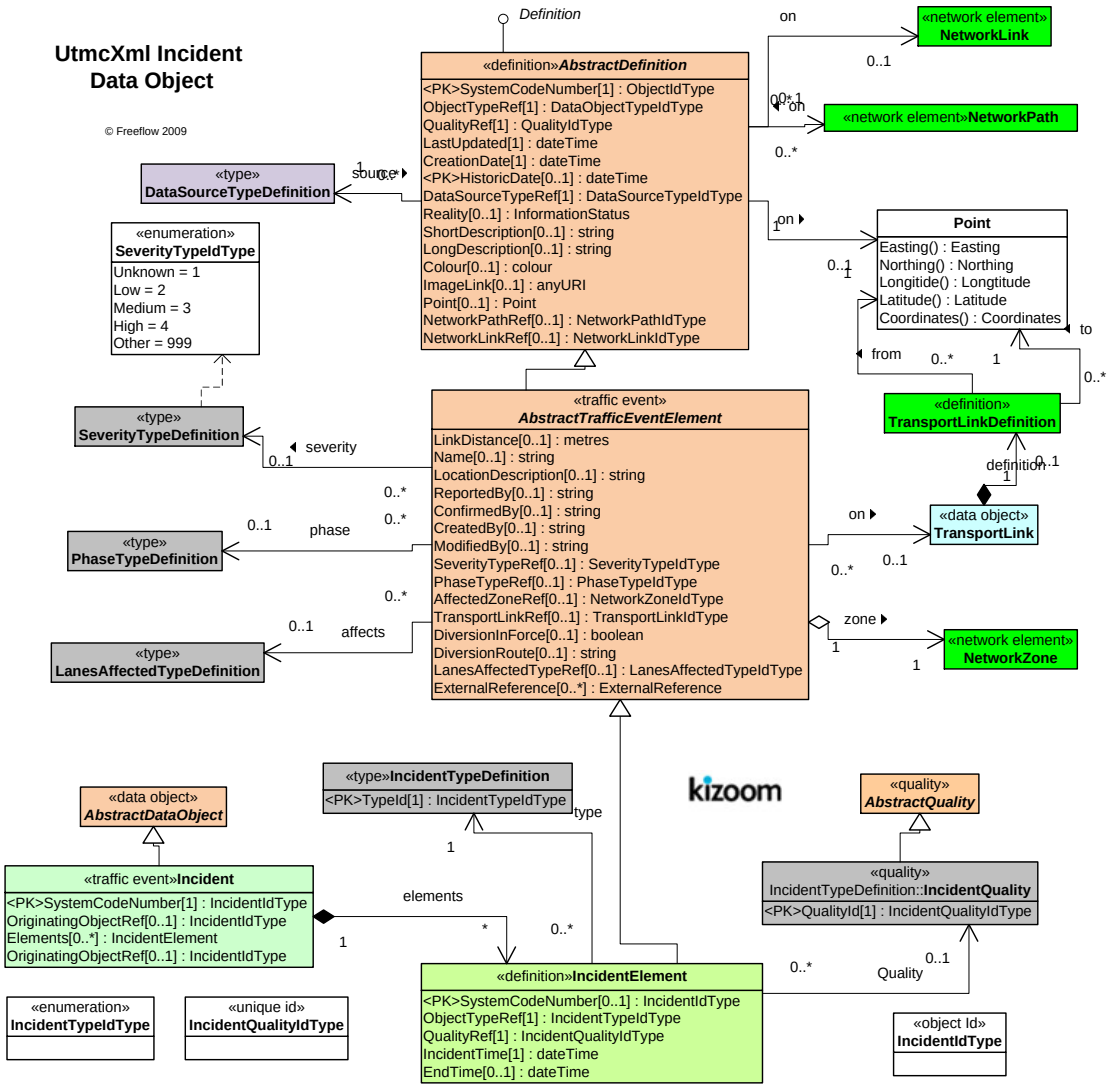


Figure 5-28 UML diagram of UTMC Incident - Details

© Copyright Kizoom 2007-2009

UTMC EVENT

UTMC EVENT – OVERVIEW

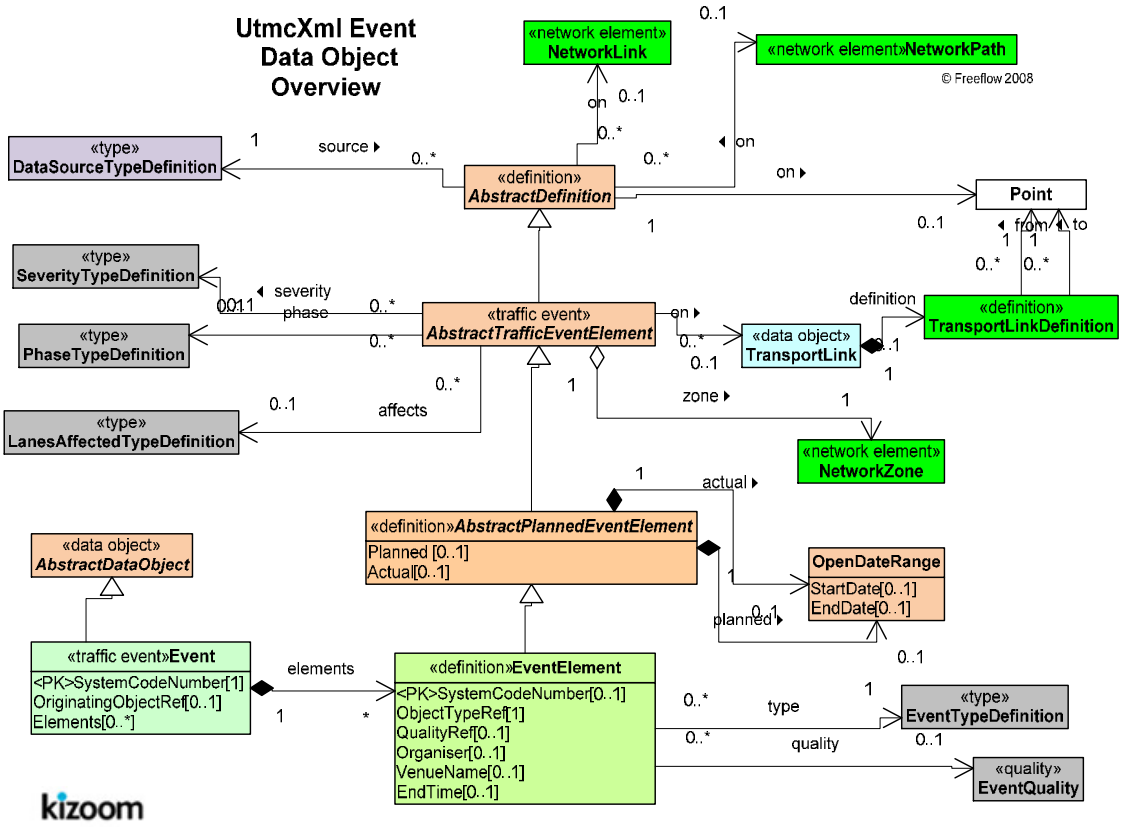


Figure 5-29 UML diagram of UTMC Event - Overview

© Copyright Kizoom 2007-2009

UTMC EVENT – DETAILS

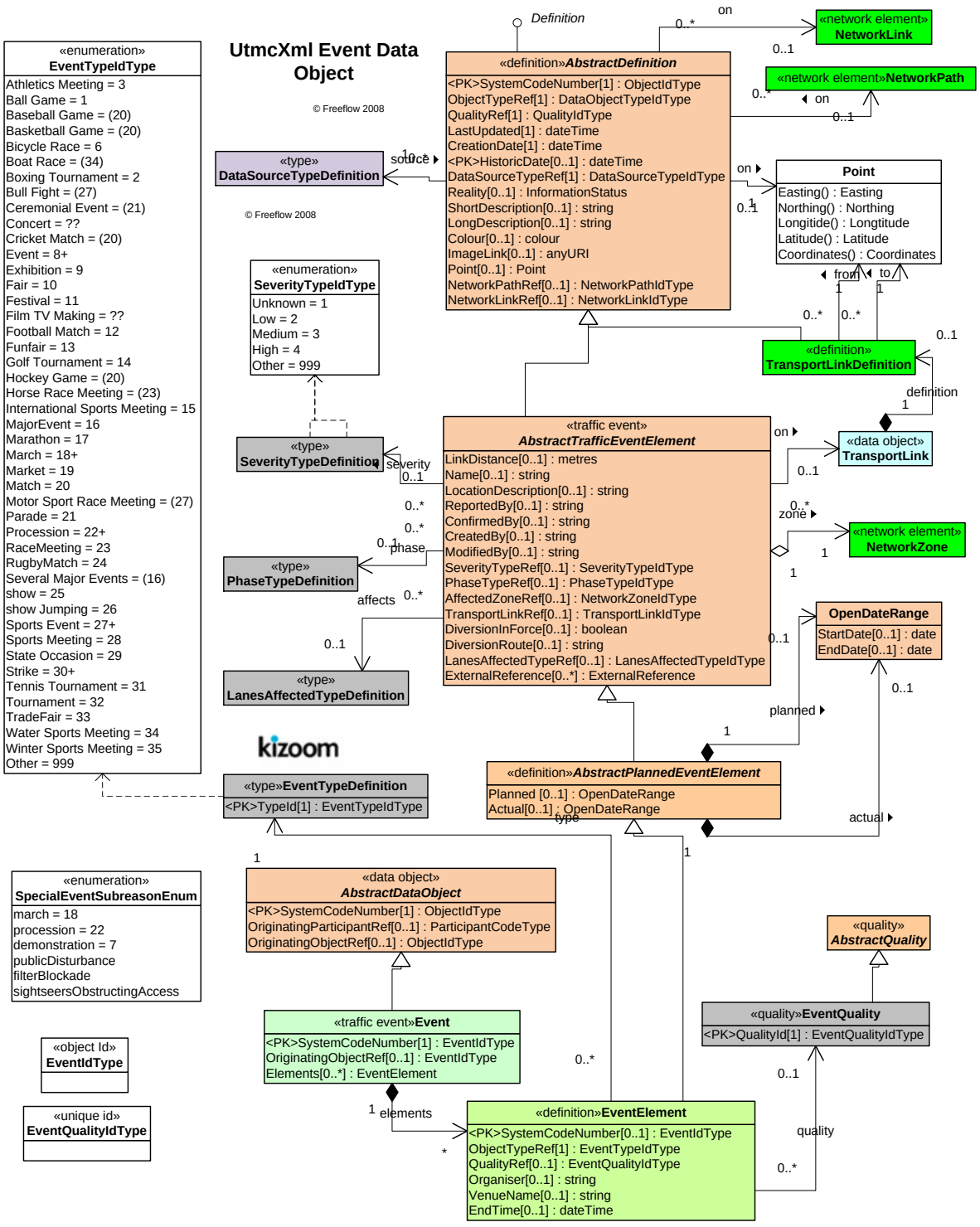


Figure 5-30 UML diagram of UTMC Event - Details

© Copyright Kizoom 2007-2009

UTMC ROADWORKS

UTMC ROADWORKS – OVERVIEW

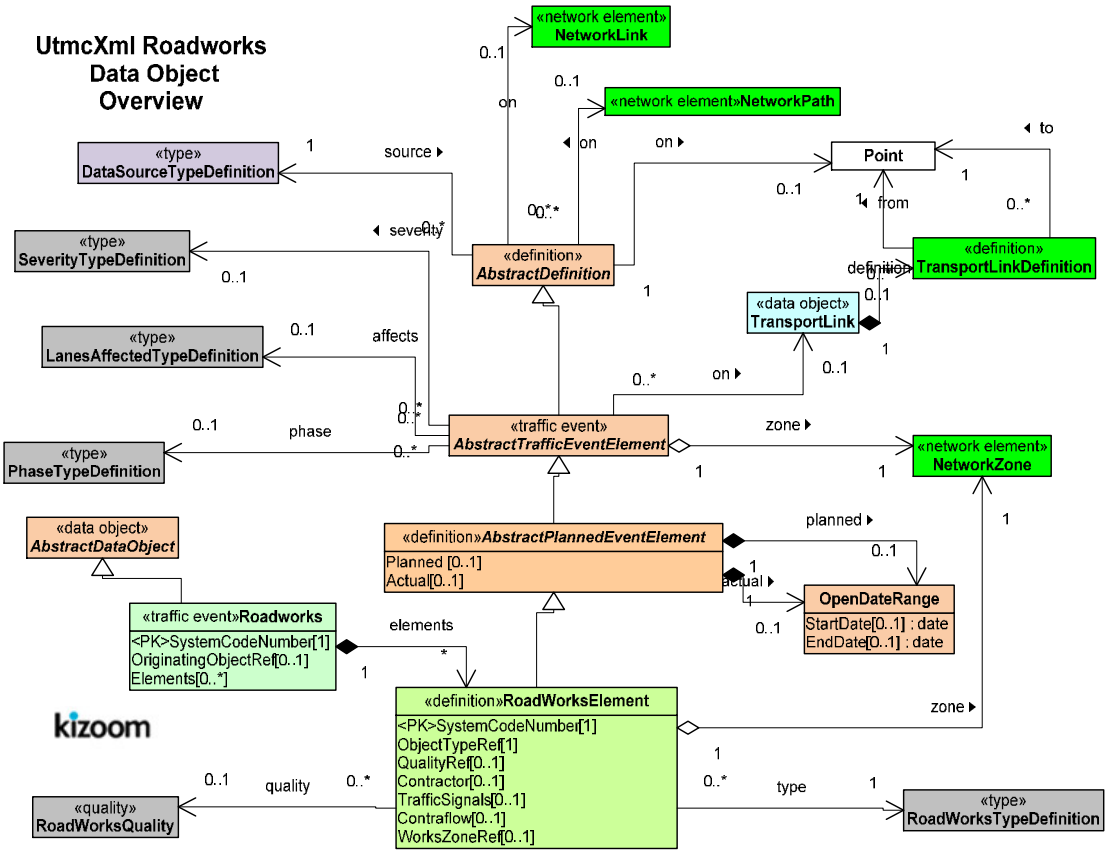


Figure 5-31 UML diagram of UTMC Roadworks - Overview

© Copyright Kizoom 2007-2009

UTMC ROADWORKS – DETAILS

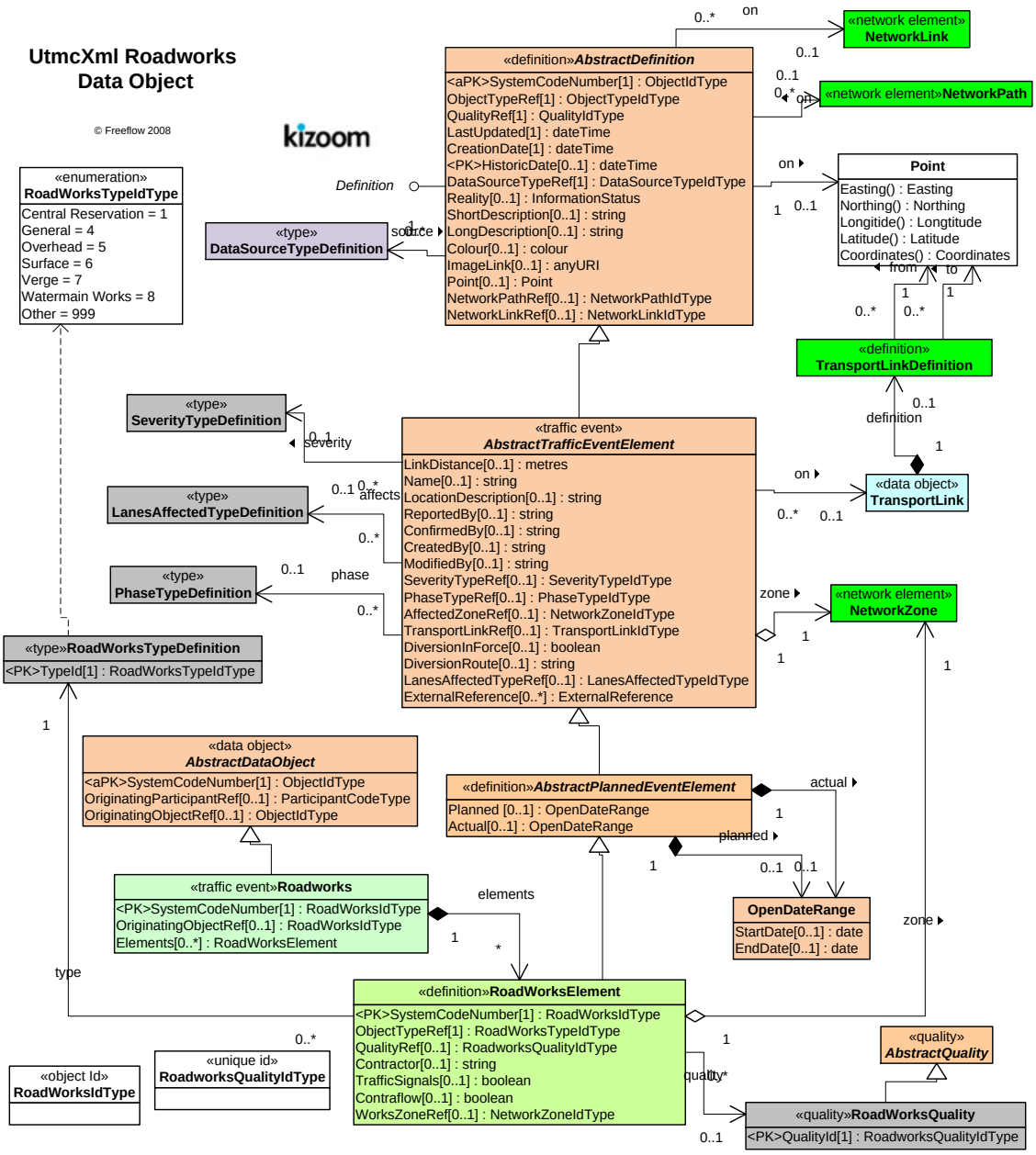


Figure 5-32 UML diagram of UTMC Roadworks - Details

© Copyright Kizoom 2007-2009

PREDICTION & PROFILE OBJECTS

Overview

UTMC Prediction Profile objects represent historic or predicted values for a time series. Each prediction or profuiles Share a common pattern of a Header object plus detailed Data elements.

MORE TO ADD

ABSTRACT PREDICTION & PROFILE MODEL – QUANTISED OBJECT

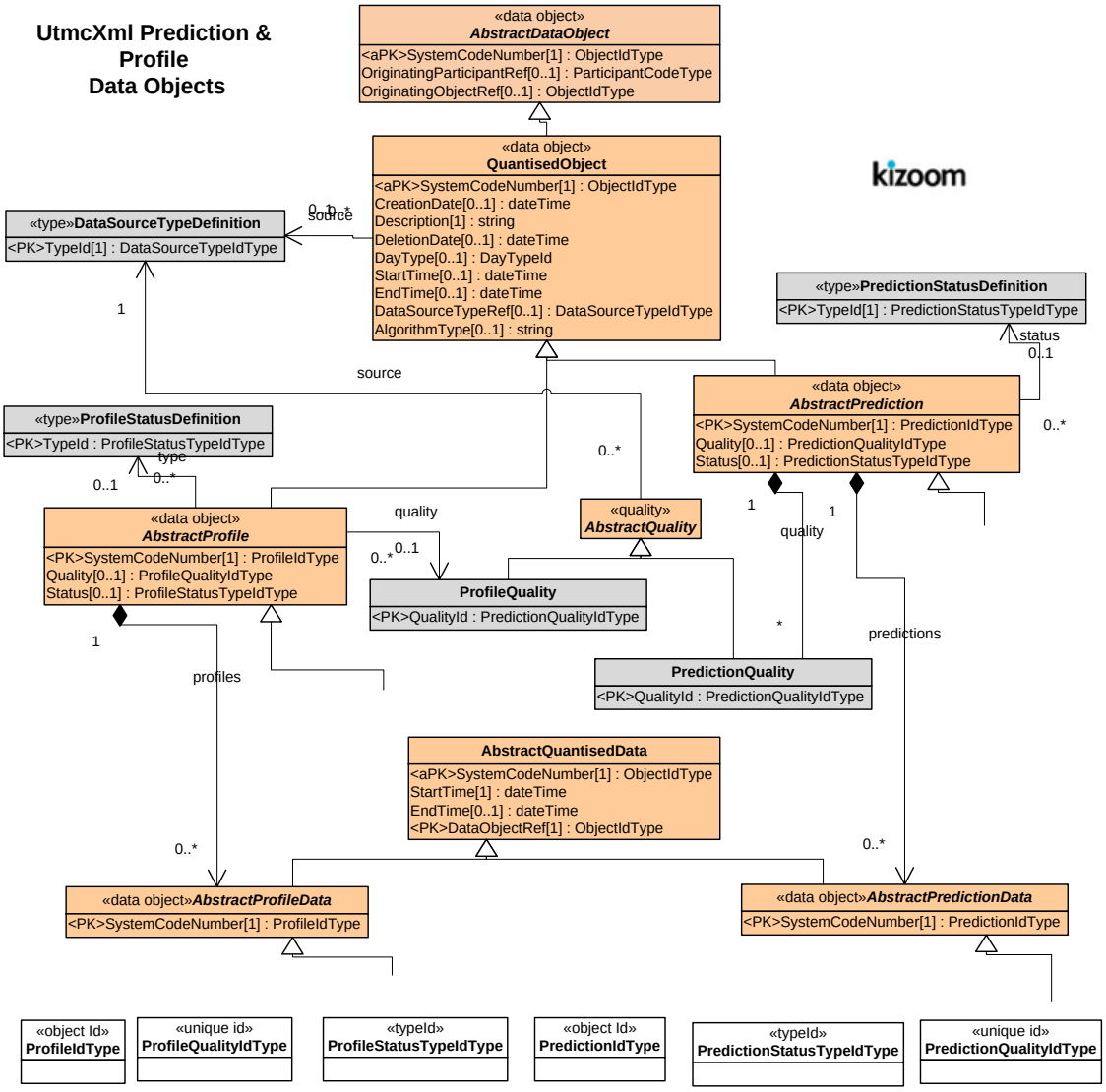


Figure 5-33 Abstract UTMC Prediction & Profile Objects

© Copyright Kizoom 2007-2009

6. USE OF UTMC XML IN Bulk Publication

WHAT IS A UTMC XML PUBLICATION?

The simplest use of the UtmcXML is for outputting one or more UTMC objects as XML in a file. The file may be exchanged my any means, e.g. FTP, email or CD, or even by http request response. The UTMC XML schema can be used to support such exchange.

- UTMC Objects of any type can be included in a file.
- Entire objects or just subcomponents can be included.
- The criteria used to extract the file can be described by a **Filter** object. There are distinct filter objects for each major Data Object type (Network Objects, Device Objects, Traffic Events, etc)
- Reference data such as Type Definitions for Object Types etc can also be exchanged.
- Header data on the publication can indicate when the data was generated and what it represents (e.g. "London Traffic events", M1 traffic link flows between two times etc)

TYPICAL USE CASES

- To configure and describe the output of a system that periodically output a file with selected items from a UTMC database. For example the traffic events in an area, or the transport links current flow values.
- To exchange all the data from a UTMC data to set up a system or for archive, analysis or other purposes.
- To provide historical extracts.

FILE CONTENT

A bulk publication file will typically contain

- A ***PublicationRequest***, specifying a filter or filters, describing the selection criteria used to output the file and when it was generated. Some of these selection criteria are generic to all UTMC objects, some are specific to certain types of UTMC Data Object.
- ***PublicationDelivery***, comprising a collection of data object serialised as XML. Each Data object may contain one or more components, such as a definition, configuration etc, depending on its type. Common properties of the object such as its identifier need be supplied just once.

© Copyright Kizoom 2007-2009

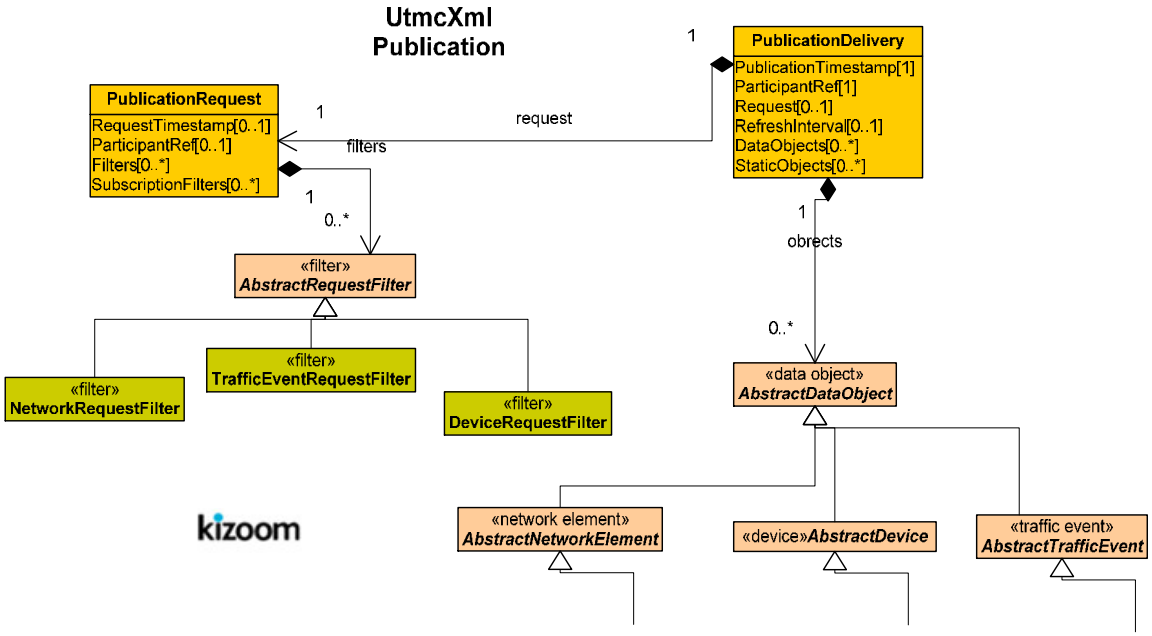


Figure 6-1 UML Diagram of a Publication.

Figure 6-2 shows further details about a UtmcXmlPublication.

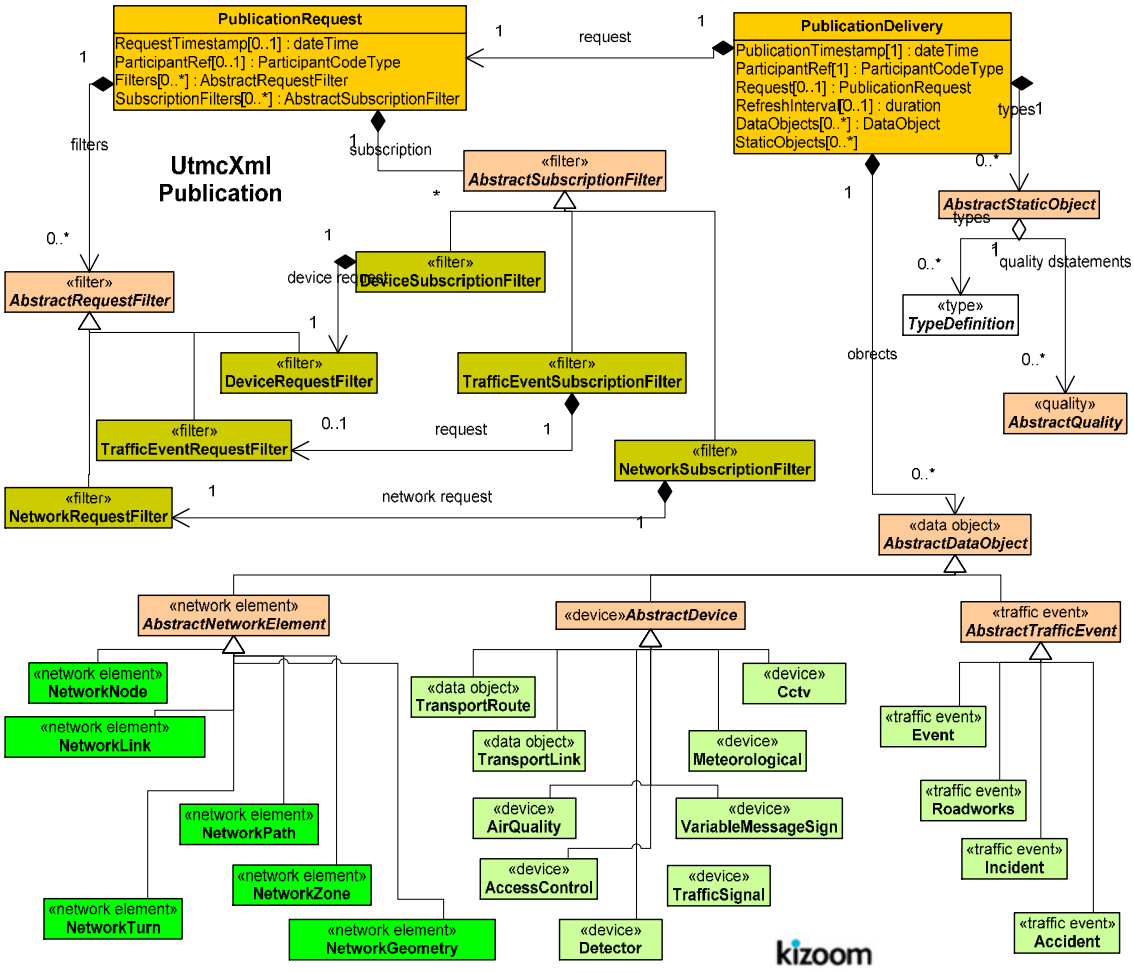


Figure 6-2 UtmcXml Publication - Details

© Copyright Kizoom 2007-2009

FILTERS

A Filter describes the contents that are to be or have been published, for example the types of objects to include, their geographical scope, and their temporal currency. Filtres can be used to create UTMC publications, and also on UTMC SIRI requests.

FILTER EXAMPLE

Filters may both used in request, and be reflected back in published content to describe the output. For example the following shows a filter requesting traffic events ofr an aria of particular severity.

```
< PublicationRequest version="1.0" xsi:schemaLocation="http://www.utmc.uk.com/utmc ../../utmc_publication.xsd"
xmlns="http://www.utmc.uk.com/utmc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                        <RequestTimestamp>2001-12-17T09:30:47.0Z</RequestTimestamp>
                        <ParticipantRef>Sys22</ParticipantRef>
                        <Filters>
                                <TrafficEventRequestFilter>
                                        <Topics>
                                                <Current/>
                                                <ZonesAffected>
                                                        <NetworkZoneRef>Zone234</NetworkZoneRef>
                                                </ZonesAffected>
                                                <SeverityTypeRef>2</SeverityTypeRef>
                                                <Types>
                                                        <AllTrafficEventTypes/>
                                                </Types>
                                        </Topics>
                                        <Policies>
                                                <DataRequestType>elements</DataRequestType>
                                        </Policies>
                                </TrafficEventRequestFilter>
                        </Filters>
</PublicationRequest>
```

The following shows the same request filter echoed back in a delivery with selected traffic events that meet the criteria.

```
<PublicationDelivery version="1.0" xsi:schemaLocation="http://www.utmc.uk.com/utmc ../../utmc_publication.xsd"
xmlns="http://www.utmc.uk.com/utmc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                <PublicationTimestamp>2001-12-17T09:30:47.0Z</PublicationTimestamp>
                <ParticipantRef>Sys123</ParticipantRef>
                <PublicationRequest version="1.0">
                        <RequestTimestamp>2001-12-17T09:30:47.0Z</RequestTimestamp>
                        <ParticipantRef>Sys22</ParticipantRef>
                        <Filters>
                                <TrafficEventRequestFilter>
                                        <Topics>
                                                <Current/>
                                                <ZonesAffected>
                                                        <NetworkZoneRef>Zone234</NetworkZoneRef>
                                                </ZonesAffected>
                                                <SeverityTypeRef>2</SeverityTypeRef>
                                                <Types>
                                                        <AllTrafficEventTypes/>
                                                </Types>
                                        </Topics>
                                        <Policies>
                                                <DataRequestType>elements</DataRequestType>
                                        </Policies>
                                </TrafficEventRequestFilter>
                        </Filters>
                </PublicationRequest>
                <PublicationRefreshInterval>P1Y2M3DT10H30M0S</PublicationRefreshInterval>
```

© Copyright Kizoom 2007-2009

```
        <!-- A collection of traffic events -->
        <DataObjects>
                <Incident>
                        <SystemCodeNumber>INc00001234</SystemCodeNumber>
                        <Elements>
                                <IncidentElement>
                                        <ObjectTypeRef>456</ObjectTypeRef>
                                        <QualityRef>54</QualityRef>
                                        <CreationDate>2009-04-01T09:30:47.0Z</CreationDate>
                                        <DataSourceTypeRef>56</DataSourceTypeRef>
                                        <Reality>real</Reality>
                                        <ShortDescription>Shagup on the M99</ShortDescription>
                                        <LongDescription>Shagup on the M99</LongDescription>
                                        <Point >
                                                <Easting>999999</Easting>
                                                <Northing>1999999</Northing>
                                        </Point>
                                        <NetworkPathRef>a</NetworkPathRef>
                                        <TransportLinkRef>Ref2004</TransportLinkRef>
                                        <LinkDistance>12.0</LinkDistance>
                                        <LocationDescription>Two milesalong the M99 past t
J12</LocationDescription>
                                        <Name>Jam</Name>
                                        <ReportedBy>Mr Keen</ReportedBy>
                                        <ConfirmedBy>Mr Eager</ConfirmedBy>
                                        <ModifiedBy>Ms Right</ModifiedBy>
                                        <AffectedZoneRef>234</AffectedZoneRef>
                                        <ExternalReference>
                                                <CountryCode>uk</CountryCode>
                                                <SystemCode>HA45</SystemCode>
                                                <SituationIdentifier>HA12245</SituationIdentifier>
                                                <SituationVersion>001</SituationVersion>
                                        </ExternalReference>
                                        <IncidentTime>2009-04-01T09:30:47.0Z</IncidentTime>
                                </IncidentElement>
                        </Elements>
                </Incident>

                <Accident>
                        <SystemCodeNumber>Acc00001234</SystemCodeNumber>
                        <Elements>
                                <AccidentElement>
                                        <ObjectTypeRef>456</ObjectTypeRef>
                                        <QualityRef>23</QualityRef>
                                        <CreationDate>2009-04-01T09:30:47.0Z</CreationDate>
                                        <DataSourceTypeRef>54</DataSourceTypeRef>
                                        <Reality>real</Reality>
                                        <ShortDescription>Crash in Scunthorpe centre</ShortDescription>
                                        <LongDescription>Crash in Scunthorpe centre</LongDescription>
                                        <Point id="NMTOKEN" >
                                                <Easting>999999</Easting>
                                                <Northing>1999999</Northing>
                                        </Point>
                                        <NetworkPathRef>a</NetworkPathRef>
                                        <LocationDescription>String</LocationDescription>
                                        <Name>normalizedString</Name>
                                        <ReportedBy>Mr Keen</ReportedBy>
                                        <AccidentTime>2009-04-01T09:30:47.0Z</AccidentTime>
                                <EndTime>2009-04-01T12:30:47.0Z</EndTime>
                                </AccidentElement>
                        </Elements>
                        <Extensions/>
                </Accident>
        </DataObjects>
</PublicationDelivery>
```

TYPES OF FILTER

There are three main types of filters in UTMC. All share a number of common features.

- Network Object Request Filter: specifies which Network Objects are to be selected.

© Copyright Kizoom 2007-2009

- Device Request Filter – For requesting Device Data objects
- Traffic Event Filter – For requesting traffic events

Each Filter has two parts, mostly optional

- **Topic** – Specifies the objects that are to be selected as a list of selection criteria that will depend on attributes of the objects to be selected. The abstract device filter allows filterting by source, network path, currency
- **Policy** – Specifies additional parameters as to "how" the selected data is to be processed, for example the level of detail for each object.

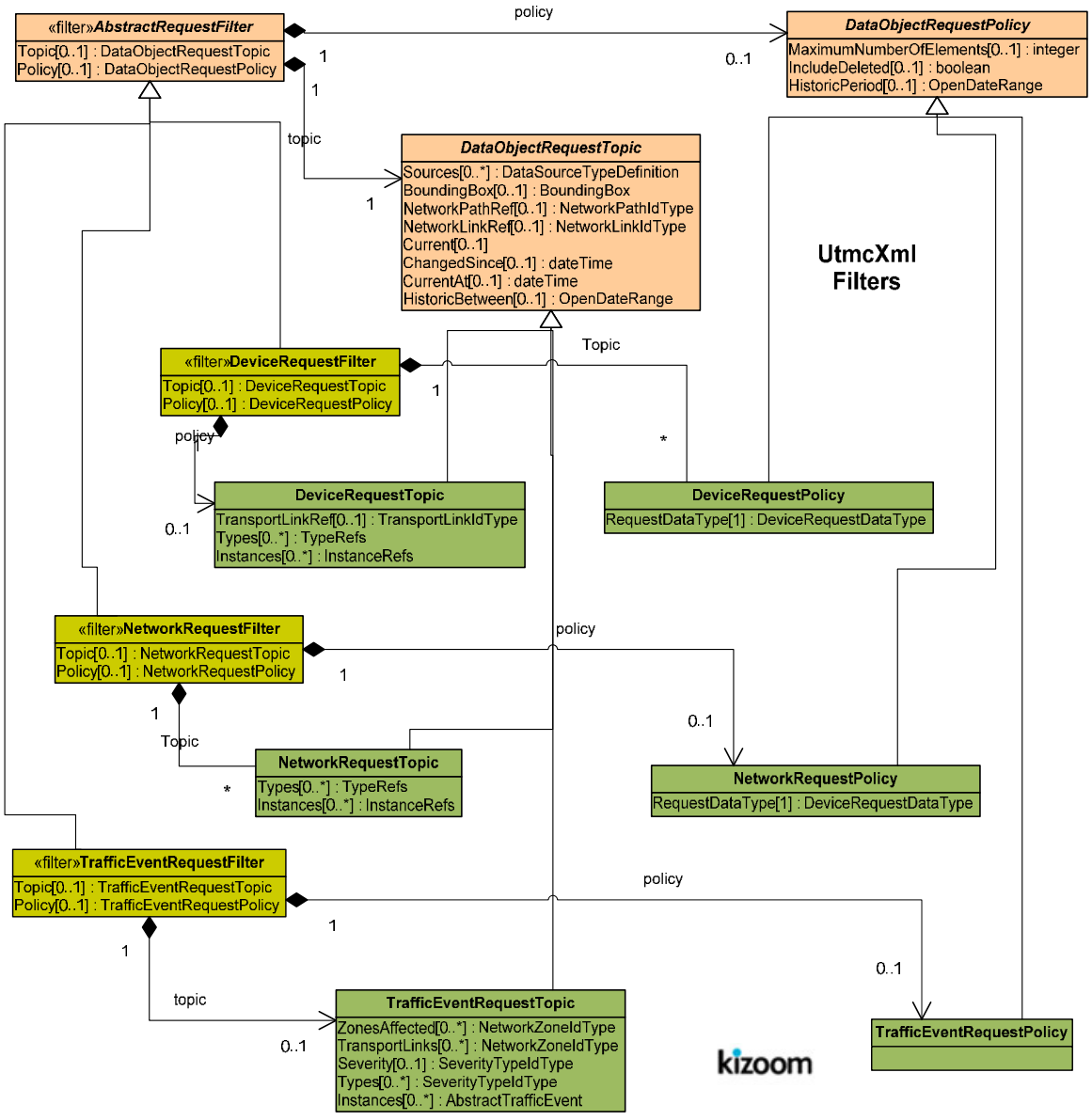


Figure 6-3 Types of Filter

© Copyright Kizoom 2007-2009

UTMC DEVICE REQUEST FILTER

The device Request Filter is used to obtain device objects. It allows additional filtering by

- Object Type (All Device Instances, All Detector Instances etc
- Object Subtype Specific types of CCTV, VMS, etc)
- Which components are to be returned: history, faults etc.

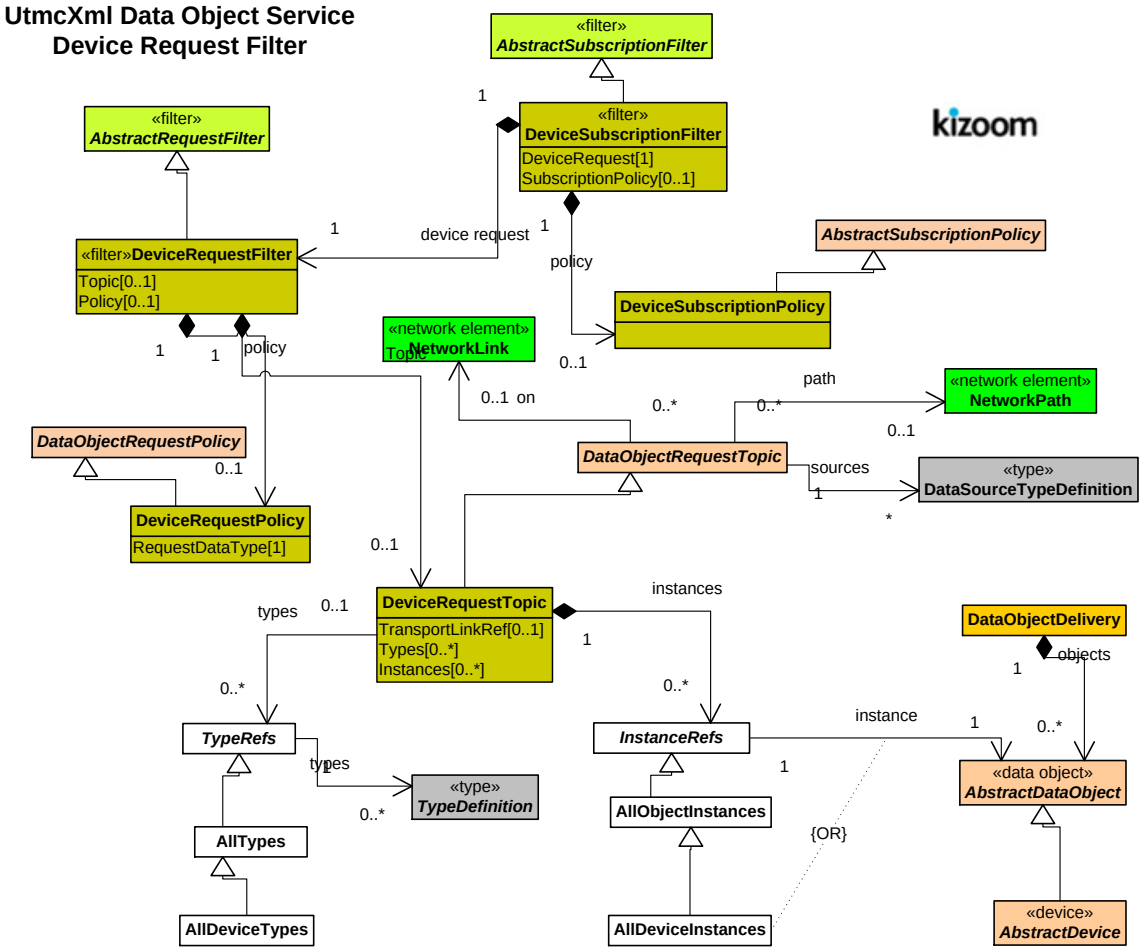


Figure 6-4 Device Rewquest Filter - Overview

© Copyright Kizoom 2007-2009

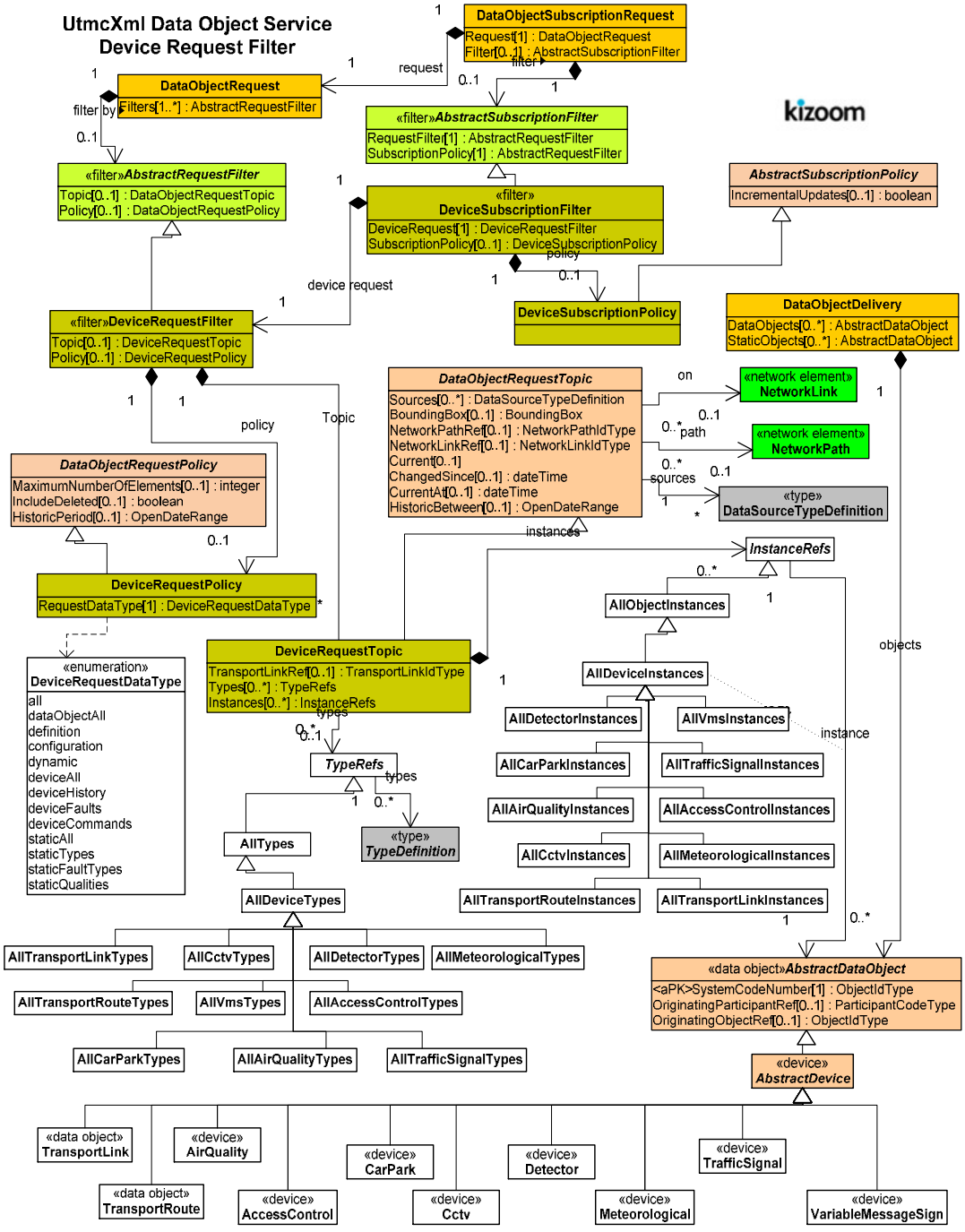


Figure 6-5 UML diagram of Device Filter

© Copyright Kizoom 2007-2009

UTMC TRAFFIC EVENT REQUEST FILTER

The Traffic Event Filter is used to obtain Traffic Event objects. It allows additional filtering by

- Object Type (All Accident Instances, All Event Instances, etc
- Object Subtype Specific types of Accident, Roadworks, etc)
- Severity & Phase, Lanes affected.

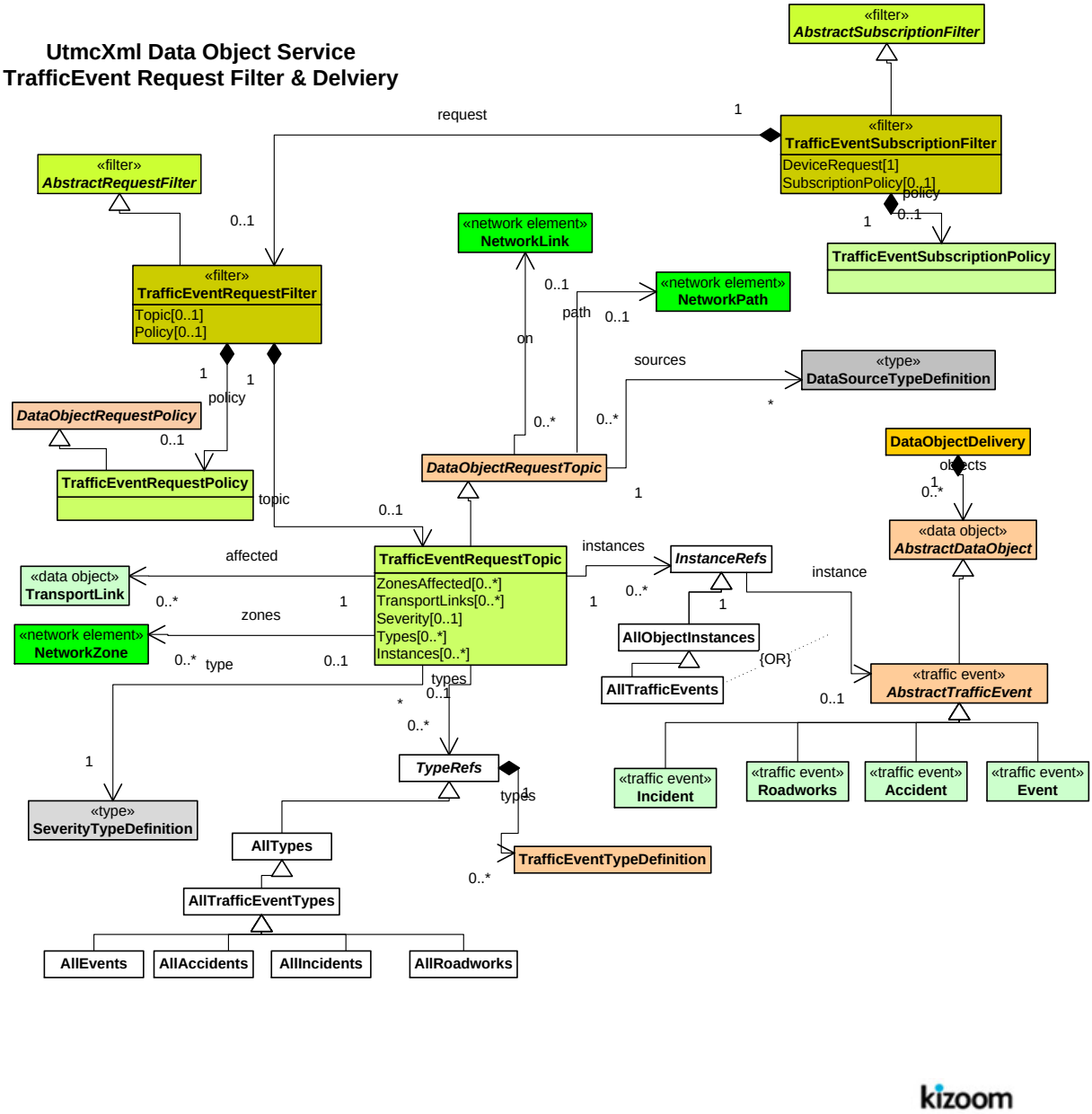


Figure 6-6 Traffic Event Request Filter - Overview

© Copyright Kizoom 2007-2009

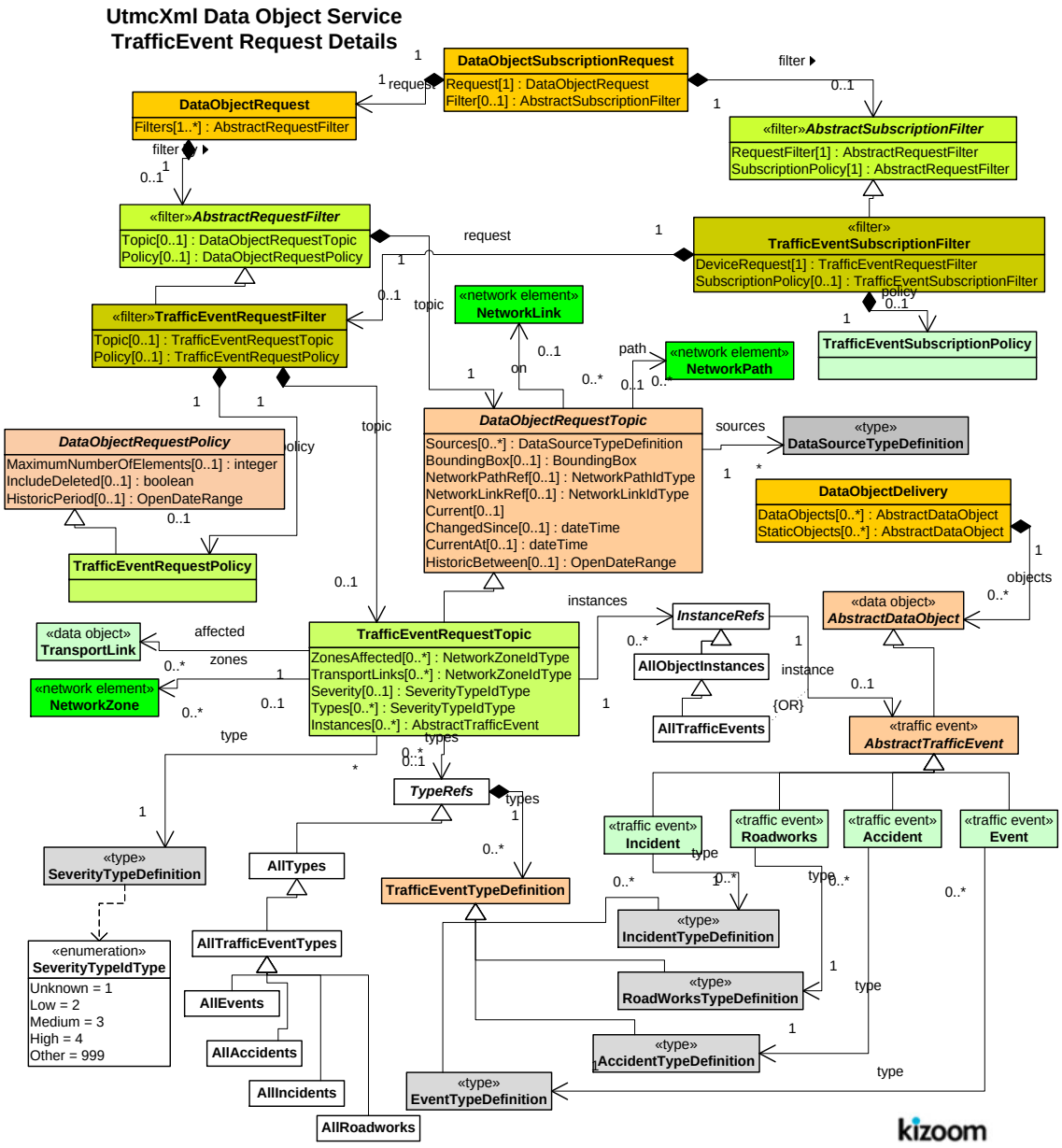


Figure 6-7 UML Diagram of Traffic Event Filter - details

© Copyright Kizoom 2007-2009

UTMC NETWORK REQUEST FILTER

The Network Filter is used to obtain UTMC Network objects. . It allows additional filtering by

- Object Type (All Node Instances, All Link Instances etc
- Object Subtype Specific types of Network Path, Network ZOne, etc)

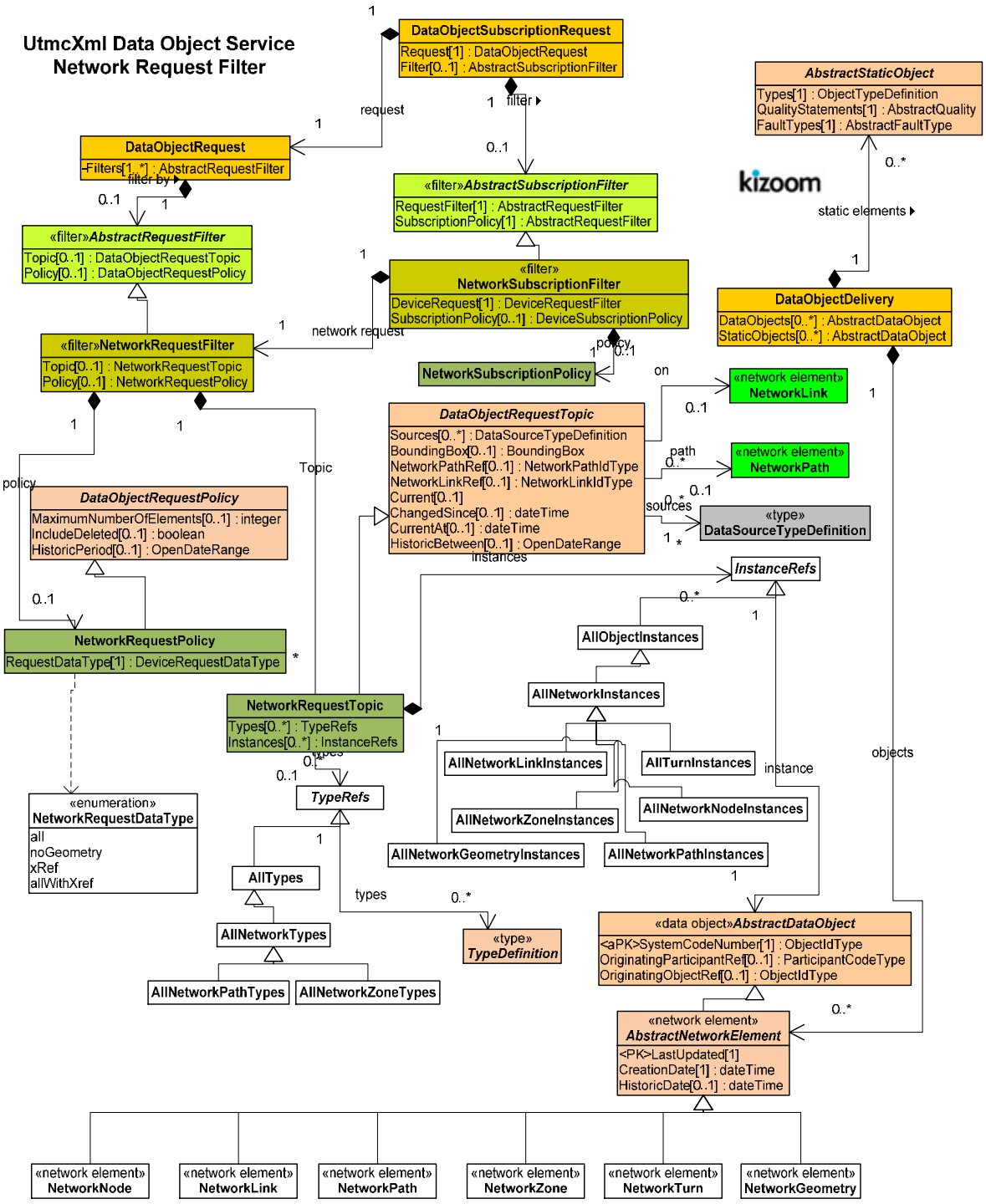


Figure 6-8 UML Diagram of Network Filter

© Copyright Kizoom 2007-2009

UTMC PREDICTION REQUEST FILTER

The Prediction Filter is used to obtain UTMC Prediction objects. . It allows additional filtering by

- Future Period covered by Prediction
- Object Instance for which prediction is for
- Prediction status

UTMC PROFILE REQUEST FILTER

The Profile Filter is used to obtain UTMC Profile objects. . It allows additional filtering by

- Profile Period c
- Object Instance for which Profile is for
- Profile status

STATIC OBJECTS

The UtmcXml StaticObject provides a means of referencing the Type Definition, Quality Statement and Fault Type instances associated with a UTMC Data Object Type . For each Static Object there is a static object , for example ***TransportLinkStaticObject***. Which groups the elements.

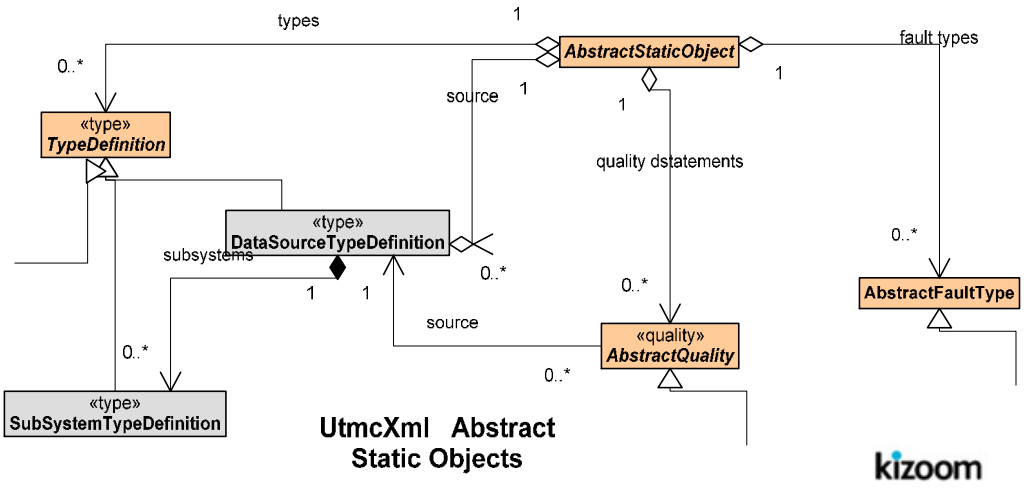


Figure 6-9 Static Objects

© Copyright Kizoom 2007-2009

TRAFFIC EVENT STATIC OBJECTS

Traffic Event Static Objects provide a way of obtaining the fixed type definitions associated with a Traffic Event

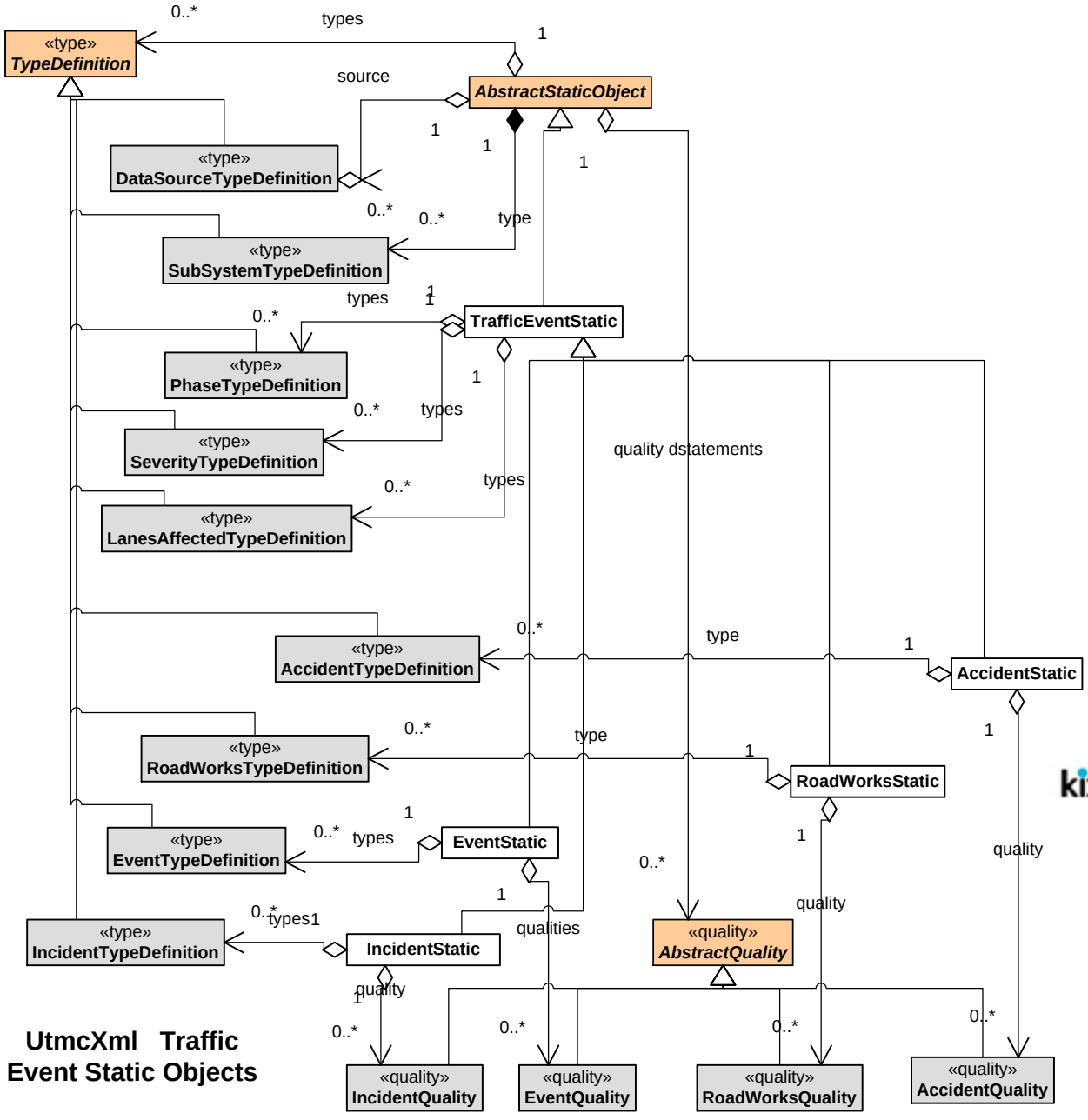


Figure 6-10 Traffic Event Static Objects

© Copyright Kizoom 2007-2009

7. DYNAMIC EXCHANGE OF UTMC XML USING SIRI

WHAT IS THE SIRI UTMC XML WEB SERVICE ?

Very often there will be a requirement to dynamically exchange UTMC data objects or parts of data objects. The UTMC XML uses the SIRI –real-time XML framework to support this. It allows the fine-grained exchange of UTMC XML data objects as XML web services, either on request/response or by publish/subscribe. Data is exchanged as XML attachments associated with http requests.

The dynamic interface is especially suitable for efficient real-time operation. Both push models with a minimum latency, and pull models, to suit Consumer responsiveness can be used.

- UTMC Objects of any type can be requested in a file.
- Entire objects or just subcomponents can be included.
- The criteria used to extract the file can be described by the same **Filter** objects that are used for publications. There are distinct filter objects for each major Data Object type (Network Objects, Device Objects, Traffic Events, etc)
- Reference data such as Type Definitions for Object Types etc can also be exchanged.
- Both "pull" (request response) and "Push" (publish/subscribe" is supported

TYPICAL USE CASES

- To dynamically send objects as they change in real-time. For example the traffic events in an area, or the transport links current flow values.
- To obtain definition & configuration objects to provision a client system.
- To obtain real-0time data on a periodic basis – e.g. travel times., accidents, roadworks
- To obtain travel times for a specific route
- To dynamically send commands back to a UTMC system
- To provide data from a UTMC database to an Intelligent Decision Support (IDS) system and to return its outputs to the UTMC database

© Copyright Kizoom 2007-2009

WHAT IS SIRI?

SIRI specifies a European data interface standard for exchanging information about the planned, current or projected performance of real-time data, such as public transport operations, between different computer systems.

- SIRI comprises a modular set of discrete **functional services** for exchanging data for specific information system process, for example stop departures, vehicle movement, etc.
- SIRI aims to incorporate of the best of various national and proprietary standards from across Europe and to deliver these as **web services** using a modern XML schema.
- All SIRI services are provided over a standardised **Communications layer**, based on a Web Services Architecture and described in Parts1 & 2 of the SIRI standard .
    - The Communications layer upholds a consistent approach for all the functional services to Security, Authentication, Version Negotiation, Recovery/Restart, and Access Control/Filtering.
    - To support different operating requirements, two main patterns of interactions are supported: an immediate **Request/Response** protocol; and an asynchronous **Publish/Subscribe** protocol. The Publish/Subscribe can be further elaborated with a fetched delivery interaction to optimise the use of bandwidth.

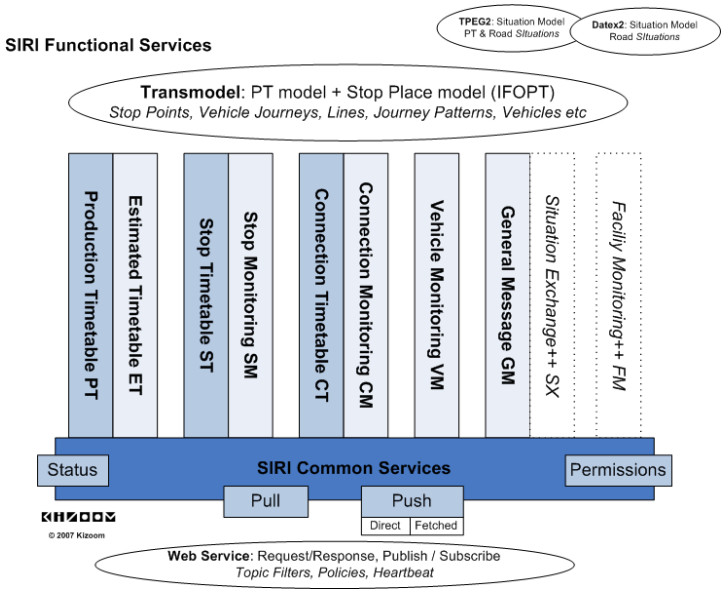


Figure 7-1 SIRI Services

- Parts 3, 4 and 5 of the SIRI standard provides a number of functional services for example for real-time stop departures. Additional services may be added that use the same Communications layer, allowing suppliers to reuse standardise their request, subscription and service management implementations. The UtmcXml functional services are an example of further additional services.

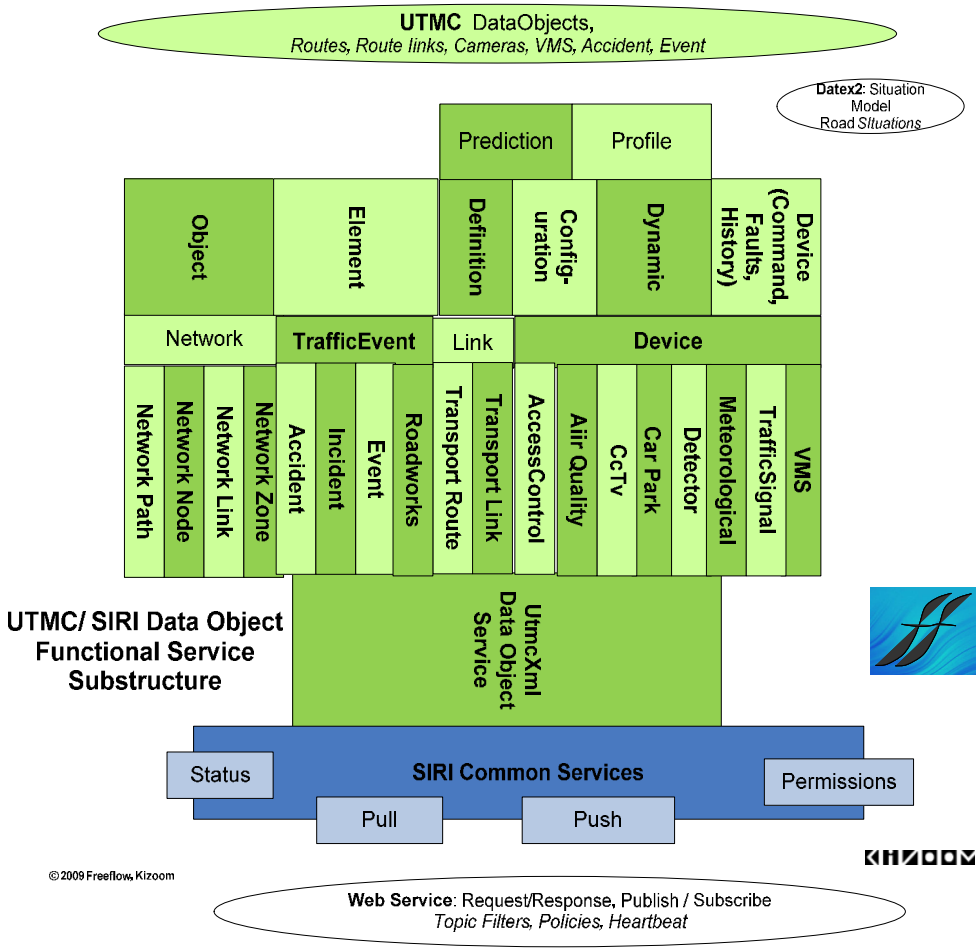
© Copyright Kizoom 2007-2009

WHAT FUNCTIONAL SERVICES ARE AVAILABLE IN SIRI-UTMC-DO ?

Different domains, such as UTMC, may add their own specific request to SIRI – (SIRI is a framework as well as a set of specific PT requests). UtmcXml currently comprises the following single SIRI **functional service** (See Figure 7-1):

- **UTMC** Data Object Request (SIRI-UTMC-DO) provides a **general purpose** request that can be used to request any UTMC Data Object or Data Object component.
    - A Filter object can be used to specify which elements are required.
    - Different levels of detail can be requested
    - The UTMC-DO request can be used both directly in request/response and in publish/subscribe interactions..

7-2 Shows the UTMC Siri service



7-2 SIRI-UTMC-DO Service

© Copyright Kizoom 2007-2009

GENERAL USE OF SIRI SERVICES

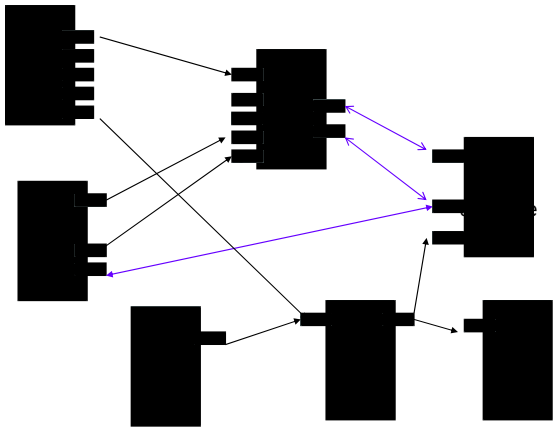
All the SIRI Functional services follow a common design pattern.

- SIRI is for the exchange of data between participant services. The server which provides the data is the ***Producer***. The server which obtains the data is the ***Consumer***.
- The same servers may sustain more than one different SIRI functional service, and support either or both Consumer and Producer roles as appropriate. The Producer service itself may be broken down into further roles (e.g. publisher, notification producer)– though a Consumer does not need to be aware of these constituent roles.
- Each functional service has unique **endpoints**, the addresses at which the service can be found.
- Each ***Participant,*** whether consumer or producer, has a unique **Participant Identifier.**
- **Payload** content is separated from message management content.
- Content may be obtained from a Producer either by an immediate direct **request /response** interaction, or by an asynchronous **publish / subscribe** interaction. In both cases the same set of request parameters are supported to specify the desired response content; for a subscription the request is wrapped in an additional **subscription request** element.
- Data exchanged by SIRI services may either be used as input to other computations, or be rendered into end user information views in any one of a number of media: e.g. **web (**HTML**,** javascript, etc**)** , **WAP (**xHTML**)**, **SMS**, **Smartphones** (j2ME, etc) nor **Voice (**VML etc**)**.

Figure 7-3 illustrates the concept of using multiple SIRI services to connect up different types of PT Information application, for example AVMS, Journey Planners, Incident Capture systems, Web delivery of stop departure and incidents, alerts, etc The UTMC DO service can similarly be used to link up different distributed machines in different configurations'

KIZOOM

Example use of SIRI Services



5

Figure 7-3 Example of use of Services

© Copyright Kizoom 2007-2009

Another common use of SIRI would be to separate the load of the real-time systems that manage and operate the public transport from the wider passenger information systems that distribute the data. Thus for example, a SIRI push service could send real-time predictions from the AVMS system to a separate service that supports passenger queries by web and mobile. This allows the latter to be scaled without affecting the reliability of the operational system.

SEPARATION OF CONCERNS

SIRI is designed to separate different software architectural concerns, in particular for communications/content and for scaling.

COMMUNICATIONS & MESSAGE TRANSPORT

SIRI separates message content from message transport related elements so that it may be used with different methods of communication, for example XML over http, XML over SOAP/WSDL. Although a default method of http is currently used, it would be possible to use the SIRI content model, with quite different transport – for example TCP IP sockets.

In order to specify an individual SIRI functional service (or indeed any web service), three different aspects of the service need to be specified; (i) the **data content** model – for SIRI based on **Transmodel**; (ii) the **Communications Transport protocol** used to serialise and exchange messages (for example, http & XML) and; (iii) the **exchange behaviour** – the rules for use of the each functional protocol as a dynamic sequence of messages subject to certain application dependent behaviour. Figure 7-4 illustrates the concept of separation of concerns. In SIRI the Data content is described as XML messages. In addition, the variables controlling the parameterised parts of the exchange behaviour are specified as XML values in a configuration or **capability** description.

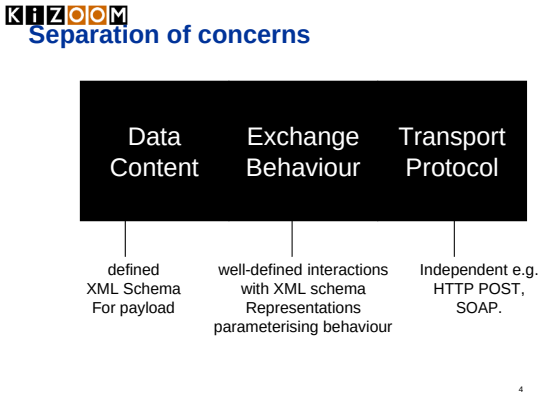


Figure 7-4 Separation of Concerns

ROLES & PATTERNS OF INTERACTION

SIRI can be regarded as a *software framework*: all the SIRI Functional Services are based on common patterns of interaction, populated with specific messages and data elements specific to a particular service. These patterns distinguish the various software roles which occur in real-time data exchange, for example Consumer, Producer, Notification Producer, Subscriber - even if these are combined on a single server for a particular implementation. The main patterns can be summarized as follows:

- Functional Service **Request Response**.

© Copyright Kizoom 2007-2009

- Functional Service **Subscription** with asynchronous **Direct Delivery**.
- Functional Service **Subscription** with asynchronous **Notification** & **Fetched Delivery.**

REQUEST/ RESPONSE FOR A SIRI FUNCTIONAL SERVICE

For the Request / Response interaction, the Consumer makes a ***Functional Service Request*** to the Producer, and receives a ***Functional Service Delivery*** message in response. The ***Functional Service Request*** may state the filtering criteria as ***Topics*** (the desired domain element references) and ***Policies*** (criteria as to how the results are to be filtered. The Topics and Policies depend on the specific functional service.

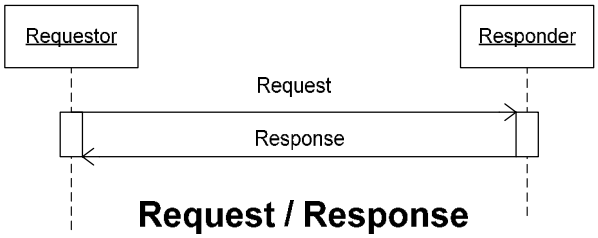


Figure 7-5 Sequence Diagram of Request Response Interaction

A Functional Service Request is always for a specific SIRI service, e.g. *Stop Monitoring, General Message*, etc. Note that the request / response pattern is also widely in SIRI for the exchange of information as pairs of messages, for example for status checking, to create subscriptions, etc.

PUBLISH/ SUBSCRIBE FOR A SIRI FUNCTIONAL SERVICE

For the Publish / Subscribe interaction, a Subscriber makes a ***Subscription Request*** to the Producer, which creates a ***Subscription*** for a Consumer (usually but not necessarily the same server as the Consumer) for the requested lease period for a specified ***Functional Service Request*** and ***Subscription Policy*** The Producer will subsequently send one or more asynchronous ***Delivery*** messages if the Request criteria are met and continue doing so in accordance with the subscription policy, up to the end of the lease. The Filtering criteria can include change thresholds or 'hysteresis' values. The Subscription will eventually expire or be terminated.

© Copyright Kizoom 2007-2009

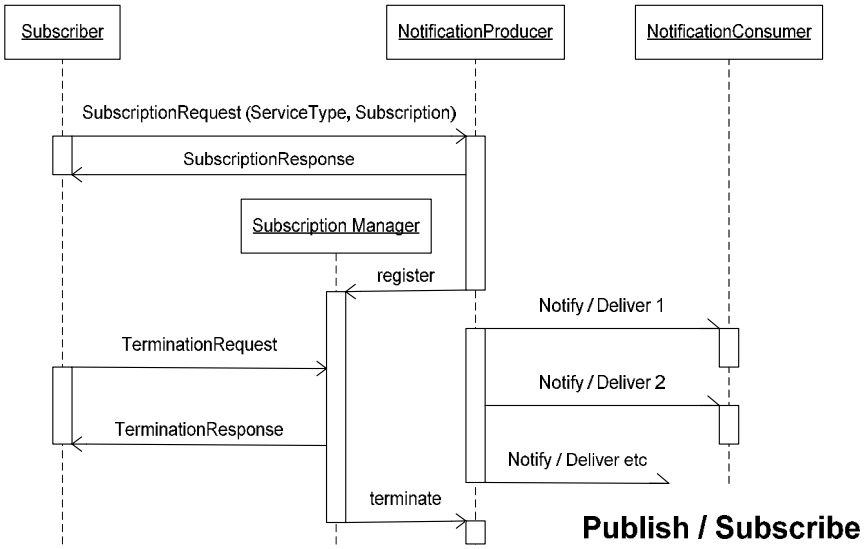


Figure 7-6 Sequence Diagram of Publish Subscribe Interaction

PUBLISH/ SUBSCRIBE WITH SEPARATE NOTIFICATION & FETCHED DELIVERY

In the simplest use, SIRI services follow of the normal web service paradigm of **direct delivery,** returning content in response to a request or a subscription. To allow additional optimisation of bandwidth and device queuing, and to provide compatibility with legacy systems, SIRI also supports an **additional** delivery variant of **fetched delivery** in which data is returned in two steps: first a **notification message** from Producer to Consumer that data is available, then a Request/ Response interaction from Consumer to Producer to fetch the data using a **delivery** message. The payload content of the delivery message is the same as for the delivery of a direct response

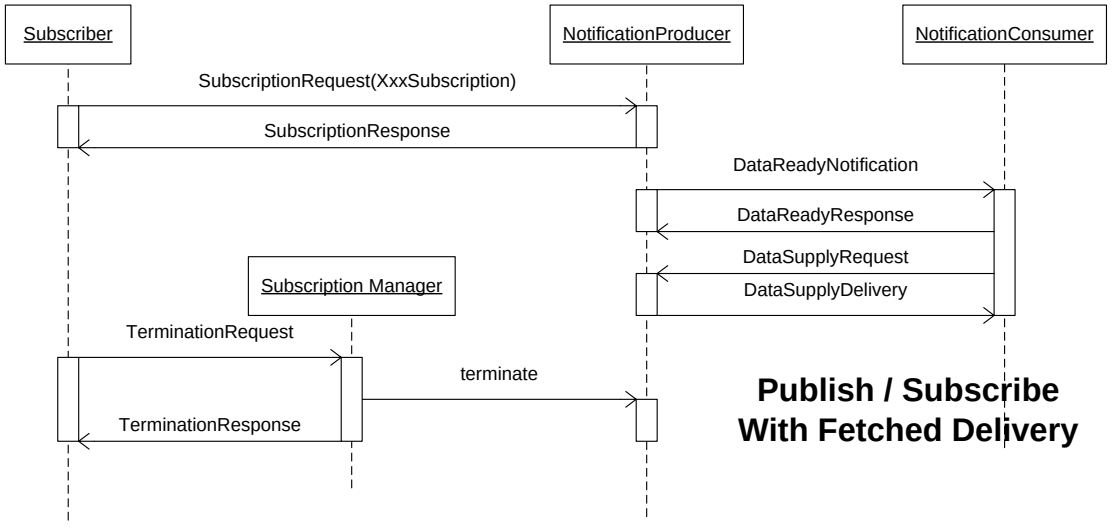


Figure 7-7 Sequence Diagram of Publish Subscribe Interaction with Fetched Delivery

© Copyright Kizoom 2007-2009

PARTICIPANT IDENTIFIERS, MESSAGE & SUBSCRIPTION MANAGEMENT

The requests and responses of SIRI functional services use a common set of header elements to track messages and subscriptions - see Figure 7-8.

- Each Participant system is given a unique identifier. This ***ParticipantCode*** is used to provide uniqueness of reference for data entities from each system. It is used as the ***RequestorRef*** on request messages and the ***ProducerRef*** on response messages.
- Each request message can be given an identifier by the requestor (***MessageIdentifier***) that is echoed back in the response (***RequestMessageRef***). This allows a requestor to match request and responses.
- The Requestor needs to be configured to know the address, i.e. url to which new requests for a SIRI functional service are to be sent. The request can contain an ***EndPointAddress*** to which responses are to be sent.
- All messages are timestamped with an identifier in UTC.

REQUEST RESPONSE ENTITIES & MESSAGE ELEMENTS

Figure 7-8 Illustrates the common elements involved in SIRI request/response interactions.

- SIRI functional service requests are sent from a ***ConsumerService*** to a ***ProducerService***, which returns a delivery which satisfies the request *Topics* and *Policies*
- A ***ServiceRequest*** contains one or more SIRI functional requests for specific functional services; each concrete SIRI functional service request (***SiriFsXXXRequest***) inherits common properties from an ***AbstractFunctionalServiceRequest.***
- A ***ServiceDelivery*** contains one or more delivery elements for a specific SIRI functional service (***SiriXxxFsDelivery***) - or ***ErrorConditions*** if responses cannot be returned. Each SIRI functional service delivery inherits common properties from an ***Abstract-FunctionalServiceDelivery. (****Note:* Subscription attributes are not populated for deliveries for request/response).
- The ***RequestMessageRef*** on the ***ServiceDelivery*** messages references the ***MessageIdentifier*** of a ***ServiceRequest*** to which it responds. The ***RequestMessageRef*** on each individual functional service delivery (***SiriXxxFsDelivery***) within the ***ServiceDelivery*** references a ***MessageIdentifier*** on the corresponding functional service request (***SiriXxxFsRequest).***

© Copyright Kizoom 2007-2009

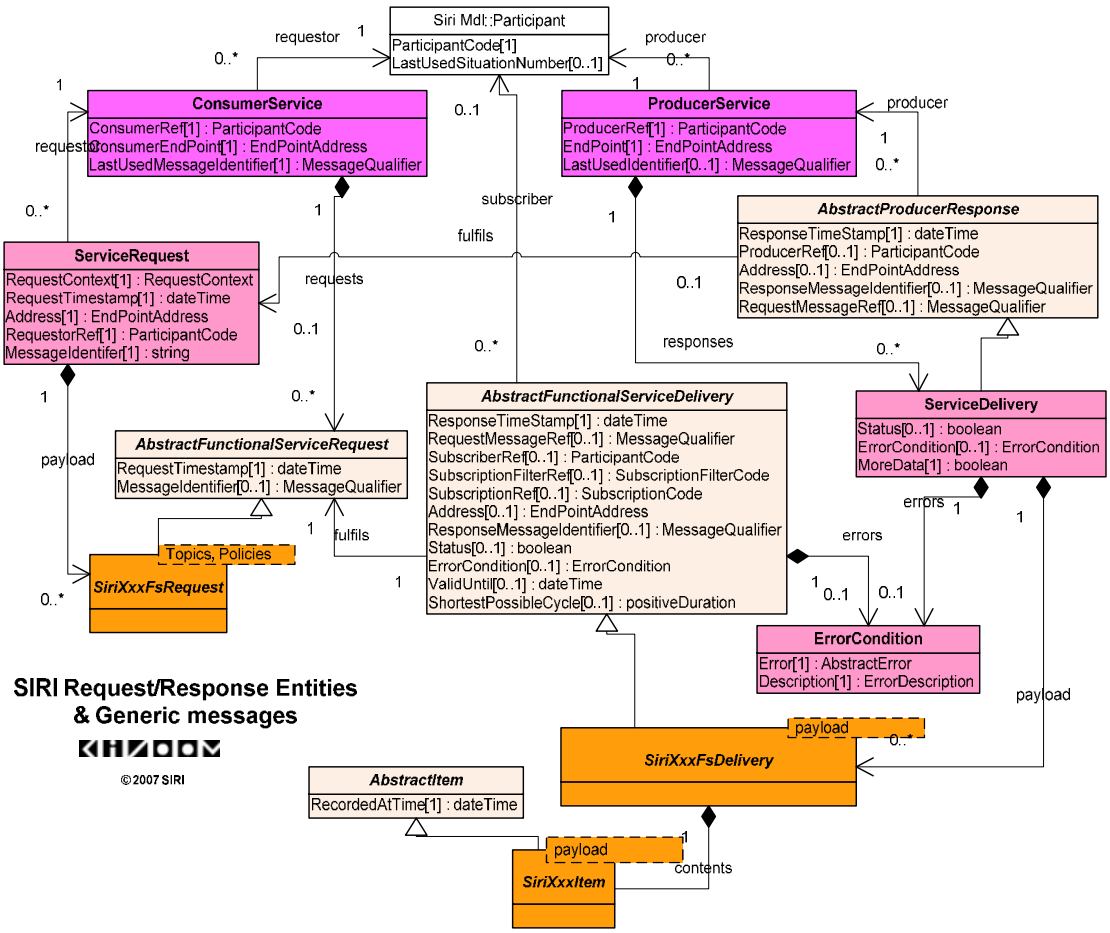


Figure 7-8 SIRI Request/Response entities

© Copyright Kizoom 2007-2009

PUBLISH SUBSCRIBE MESSAGE ELEMENTS

Figure 7-8 Illustrates the common elements involved in SIRI publish/subscribe interactions.

- The ***SubscribingService*** requests subscriptions from a **SubscriptionManager** associated with the ***ProducerService*** which creates and manages subscriptions.
- Each subscription is given a unique identifier (***SubscriptionCode)*** by the subscriber, which is specified on the subscription request for a specific SIRI functional service (***SiriXxxFs-SubscriptionRequest***) as the ***SubscriptionIdentifier,*** and then included on all deliveries sent back in response to that subscription. The identifier must be unique within the ***ParticipantCode*** of the subscriber (***SubscriberRef***).
- One or more functional service subscription requests may be submitted at a time as part of a single ***SubscriptionRequest***. All must be for the same type of service. In response to the subscription request, a ***SubscriptionResponse*** message is returned with a ***ResponseStatus*** containing the identifier of each functional service subscription (***SubscriptionRef***) and whether it has been successfully created.
- A subscription request includes an ***InitialTerminationTime*** which specifies a desired lease period. This will be echoed back as the ***ValidUntil*** time on responses.
- The ***ProducerService*** subsequently monitors the real-time feed and if the request criteria of the subscription are matched, creates deliveries that are sent by the ***ProducerService*** to the ***ConsumerService*** nominated on the subscription request. Each delivery message includes the subscription identifier (***SubscriptionRef***) and the ***ParticipantCode*** of the subscriber, termed the ***SubscriberRef***.
- The subscriber, consumer and producer service keep mirrored representations of the active subscriptions. Recovery from system failure or loss of connection is discussed separately below.
- The ***SubscribingService*** requests needs to be configured to know the address i.e. url of the system to which new subscription requests are to be sent. The request can contain a ***ConsumerEndPointAddress*** to which delivery responses are to be sent and a distinct ***SubscriberEndPointAddress*** to which an acknowledgement for the subscription request is to be sent.
- An optional capability of SIRI is to have a ***SubscriptionFilter*** that will group subscriptions from the same participant for the same type of service so that notifications and delivery data can be sent as a single message. Otherwise each subscription will give rise to separate notification and delivery messages.
- All messages are timestamped with an identifier in UTC.

© Copyright Kizoom 2007-2009

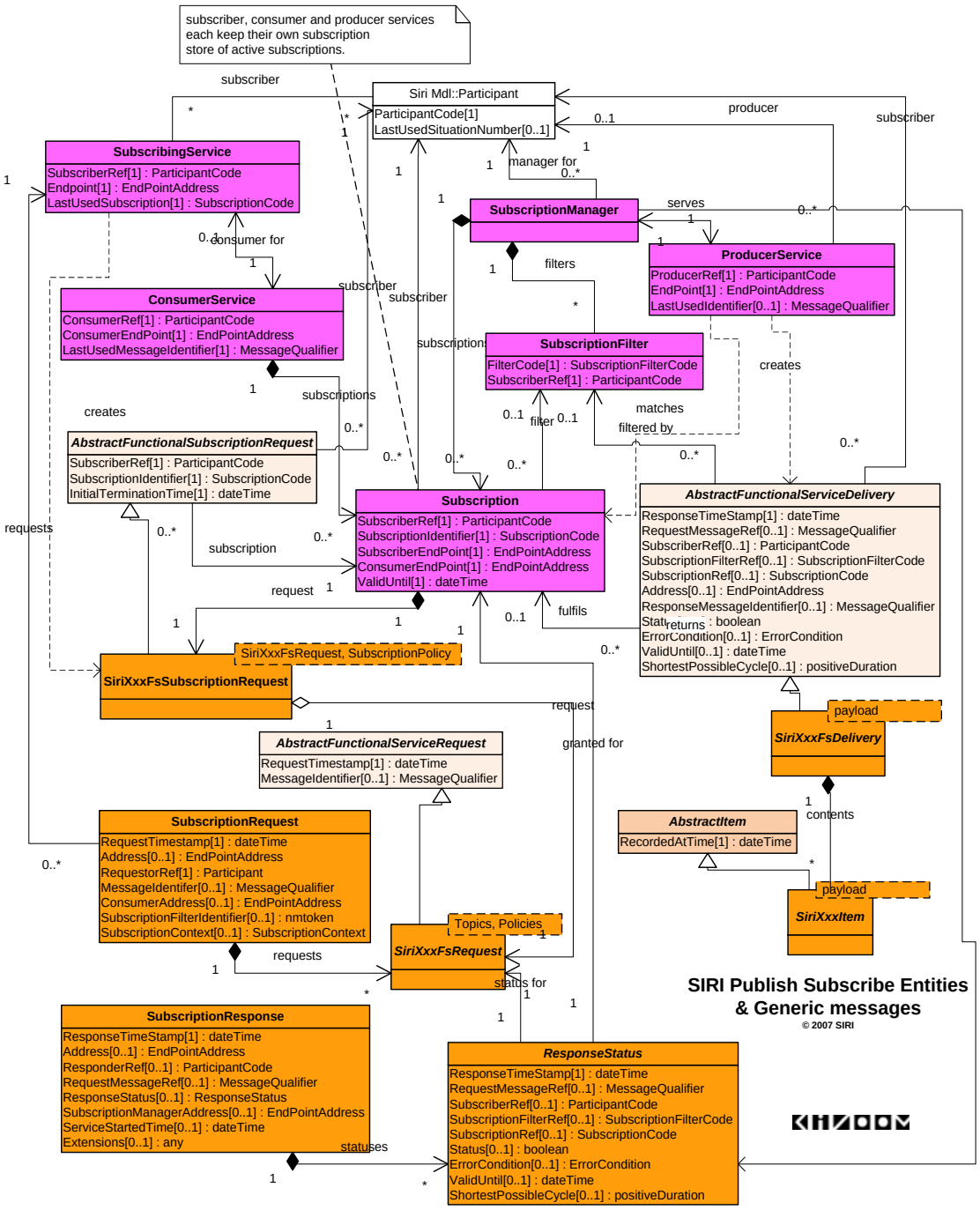


Figure 7-9 SIRI Publish/Subscribe entities

© Copyright Kizoom 2007-2009

RECOVERY & RESTART

A SIRI Publish/subscribe interaction involve a long running continuous flow of data from a Producer to a Consumer. Mechanisms are needed to allow recover from system failure of either producer, consumer or communications link. These are described in Part 2 of the SIRI specification. In brief:

A ***CheckStatus*** request can be sent between any two participants to see if the other is still working. The response includes a ***ServiceStartedTime*** which can be used to detect if the producer has failed and been restarted

A heartbeat message can be sent from Producer to Consumer at regular intervals to indicate the service is still functioning, even if there is no new real-time data. This includes a ***ServiceStartedTime*** that can be used to detect interruption of service.

it is up to the subscribing service to detect interruptions of service and if necessary to terminate and restart subscriptions.

WHAT DOES A SIRI IMPLEMENTATION REQUIRE?

The deployment of a concrete SIRI implementation entails the following:

- The choice of one or more appropriate SIRI Functional Services to meet the application needs.
- A country and or local profile that will set out **delivery method,** the **capabilities** to be supported, the **data reference system** for stop identifiers, etc.
- Allocation of unique **Participant** Identifiers to identify the different systems.
- A **Producer application** for the functional service deployed to a server at a known endpoint URL.
- A **Consumer application** for the functional service deployed to a client.
- A **Status service** to check that the service is available and working correctly and that there has not been an interruption.

© Copyright Kizoom 2007-2009

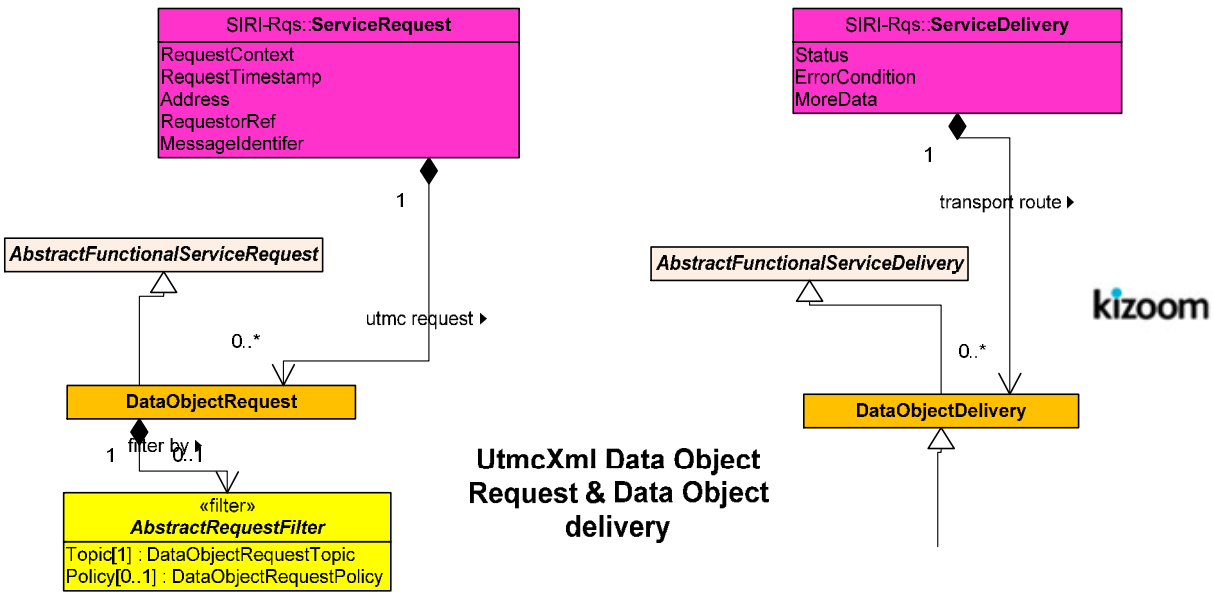
8. UTMCXML DATA OBJECT SERVICE (DO)

The UtmcXml **Data Object Service** provides a means of obtaining UTMC Data Objects and their supporting objects. Both Request/Response and Publish/Subscribe interactions may be used. It can be used by displays and other delivery services to obtain any UTMC data element., or by a UTMC database to obtain information from other services, such as UTC or IDS systems. The choice of data elements to display and the presentation format is up to the client system.

- The Request includes one or more Request Filters with which to specify which elements are to be returned.
    - The Filter *Topics* allow a Consumer system to specify which object types and which object values are required.
    - The *Filter Policies* allow a Consumer system to control the amount of data returned, both in terms of the **number of elements** and **level of detail**.
    - Different Filters may be specified for different Data Object types; for example a
- The *Subscription Policies* allow a Consumer system to control the behaviour of the subscription..
- The *Delivery* message can include not only service arrival and departure times, but also text notices and the identifiers of Situation elements associated with the stop and/or line.

UTMCXML-DO: REQUEST

The ***DataObjectRequest*** is used to request data objects as specified by a filter. A ***DataObjectDelivery*** is returned, containing one or more data objects, or error codes.



- **Figure 8-1 DataObjectRequest- Summary**

© Copyright Kizoom 2007-2009

UTMCXML-DO: SUBSCRIPTION & REQUEST

The a subscription to be sent data when it changes is created with a ***DataObjectSubscriptionRequest***. This wraps a ***DataObjectRequest***.

UTMCDATAOBJECTREQUEST SUMMARY

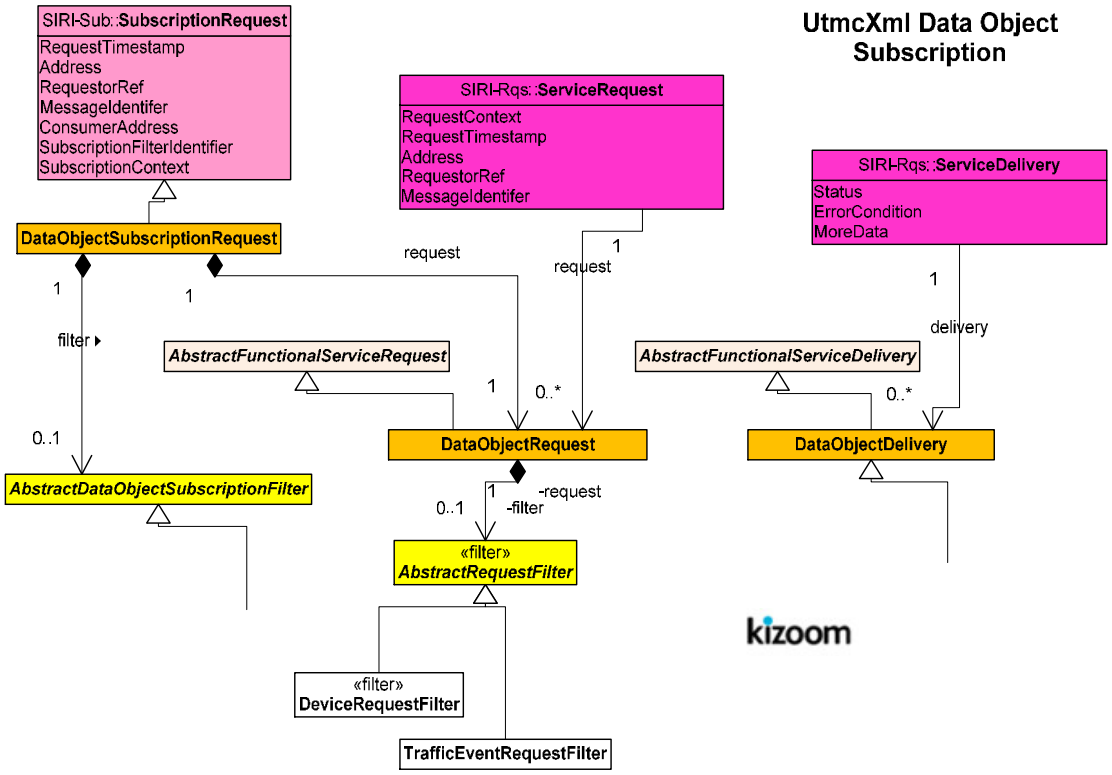


Figure 8-2 DataObjectRequest - Summary

© Copyright Kizoom 2007-2009

DATAOBJECTREQUEST DETAIL

Figure 8-3 shows a UTLC request and response in some more detail. For details of the UtmcXML representation of UTMC filter and Data objects see earlier.

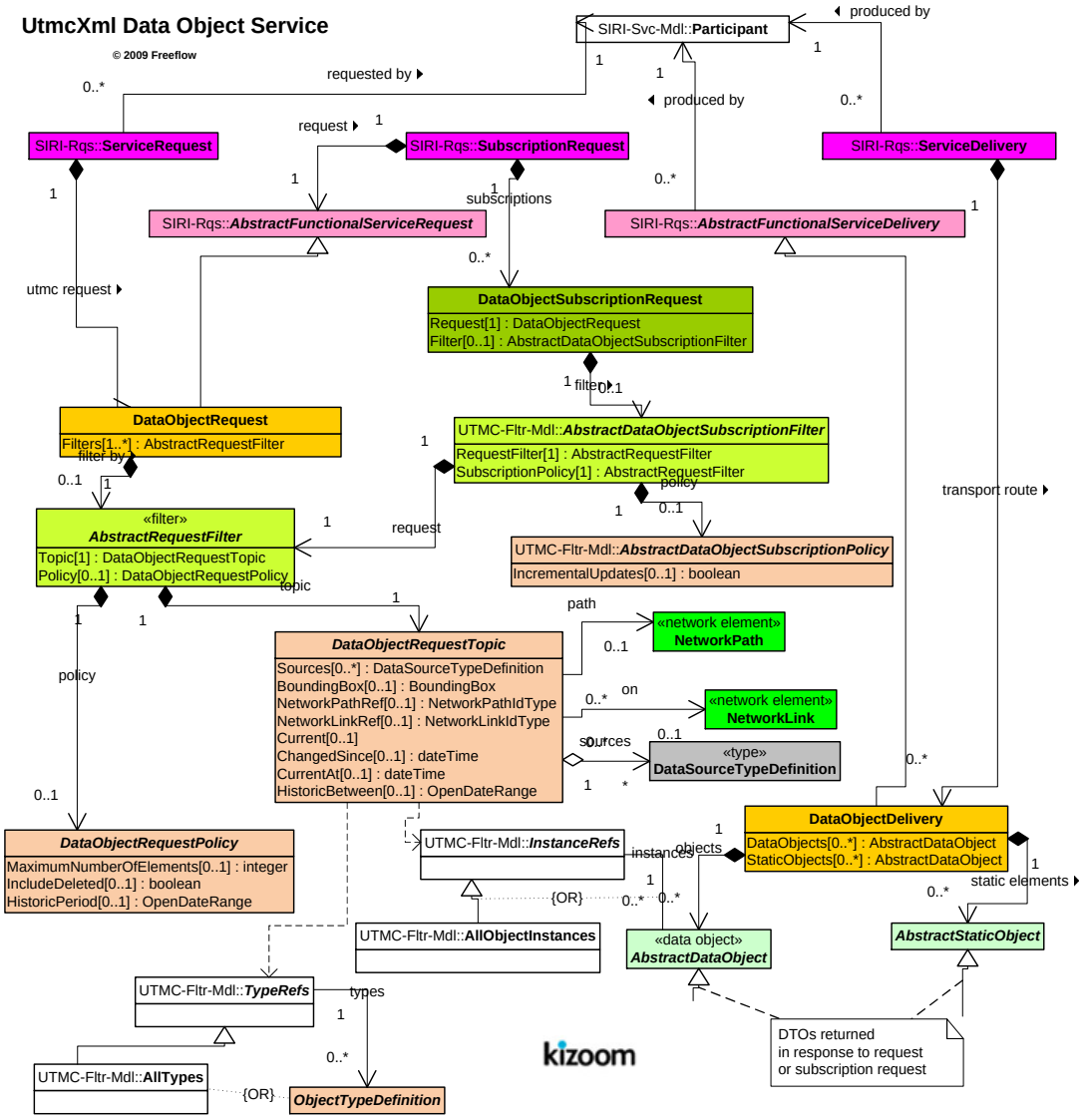


Figure 8-3 DataObjectRequest – Detail

XML EXAMPLE OF A UTMCXML FUNCTIONAL SERVICE REQUEST & DELIVERY USING SIRI

The following two XML code fragments illustrate a request/response message pair for a Functional Service - the SIRI-SM Data Object service, populated to support a simple departure board.

DATA OBJECT REQUEST EXAMPLE

© Copyright Kizoom 2007-2009

The following is an example of a ***DataObjectRequest*** for some ***TransportLink***. Instances. The request consists of a standard header, which is similar for all SIRI requests, and then a TransportLinkFilter specifying Topics and Policies that are specific to the UTMC DataObject. The request asks for the dynamic anpr data for all Links on a given network path. . .

```
<DataObjectRequest xsi:schemaLocation="http://www.siri.org.uk/siri ../../utmc.xsd" xmlns:siri="http://www.siri.org.uk/siri"
xmlns="http://www.utmc.uk.com/utmc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <siri:RequestTimestamp>2001-12-17T09:30:47.0Z</siri:RequestTimestamp>
        <siri:MessageIdentifier>http://www.altova.com</siri:MessageIdentifier>
        <Filters>
                <DeviceRequestFilter>
                        <Topics>
                                <Sources>
                                        <DataSourceTypeRef>23</DataSourceTypeRef>
                                </Sources>
                                <BoundingBox>
                                        <UpperLeft id="NMTOKEN" srsName="String">
                                                <GridType>UKOS</GridType>
                                                <Easting>999999</Easting>
                                                <Northing>1999999</Northing>
                                        </UpperLeft>
                                        <LowerRight id="NMTOKEN" srsName="String">
                                                <GridType>UKOS</GridType>
                                                <Easting>999999</Easting>
                                                <Northing>1999999</Northing>
                                        </LowerRight>
                                </BoundingBox>
                                <NetworkPathRef>a</NetworkPathRef>
                                <ChangedSince>2008-12-17T09:30:47.0Z</ChangedSince>
                                <Types>
                                        <AllDetectorTypes/>
                                </Types>
                        </Topics>
                        <Policies>
                                <MaximumNumberOfElements>500</MaximumNumberOfElements>
                                <IncludeDeleted>true</IncludeDeleted>
                                <RequestDataType>definition</RequestDataType>
                        </Policies>
                </DeviceRequestFilter>
        </Filters>
        <siri:Extensions/>
</DataObjectRequest>
```

DATA OBJECT REQUEST DELIVERY EXAMPLE

The following is an example of a ***DataObjectDelivery*** in response to a ***DataObjectRequest.*** It shows a single ***StopVisit*** for a detail level of *Normal.* The delivery consists of a standard header, which is similar for all SIRI requests, and then payload that is specific to the functional service.

```
<ServiceDelivery>
        <!--=======HEADER========================================================= -->
        <ResponseTimestamp>2004-12-17T09:30:46-05:00</ResponseTimestamp>
        <ProducerRef>KUBRICK</ProducerRef>
        <!--=======PAYLOAD======================================================== -->
        <DataObjectDelivery version="0.1d">
                <ResponseTimestamp>2004-12-17T09:30:47-05:00</ResponseTimestamp>
                <RequestMessageRef>NDR06756</RequestMessageRef>
                <!--====FIRST ARRIVAL ========================================== -->
        <Detector version="01">
                        <SystemCodeNumber>DT456</SystemCodeNumber>
                        <Definitions>
                                <DetectorDefinition>
                                        <ObjectTypeRef>67</ObjectTypeRef>
                                        <QualityRef>231</QualityRef>
                                        <CreationDate>2001-12-17T09:30:47.0Z</CreationDate>
```

© Copyright Kizoom 2007-2009

```
                    <LastUpdated>2001-12-17T09:30:47.0Z</LastUpdated>
                    <DataSourceTypeRef>9</DataSourceTypeRef>
                    <ShortDescription>a</ShortDescription>
                    <LongDescription>a</LongDescription>
                    <Point>
                        <Easting>999999</Easting>
                        <Northing>1999999</Northing>
                    </Point>
                    <NetworkPathRef>NP987</NetworkPathRef>
                    <TransportLinkRef>LTL078</TransportLinkRef>
                    <LinkDistance>87</LinkDistance>
                </DetectorDefinition>
            </Definitions>
            <Configurations>
                <DetectorConfiguration>
                    <SystemCodeNumber>a</SystemCodeNumber>
                    <ConfigurationDate>2001-12-17T09:30:47.0Z</ConfigurationDate>
                    <HistoricDate>2001-12-17T09:30:47.0Z</HistoricDate>
                    <Reality>real</Reality>
                    <ThreshholdUp>
                        <Flow>45</Flow>
                        <Occupancy>456</Occupancy>
                        <Speed>34</Speed>
                    </ThreshholdUp>
                    <ThreshholdDown>
                        <Flow>67</Flow>
                        <Occupancy>3.14159E0</Occupancy>
                        <Speed>3.14159E0</Speed>
                    </ThreshholdDown>
                </DetectorConfiguration>
            </Configurations>
            <Dynamics>
                <DetectorFlowDynamic>
                    <LastUpdated>2001-12-17T09:30:47.0Z</LastUpdated>
                    <TotalFlow>21</TotalFlow>
                    <FlowInterval>P1Y2M3DT10H30M</FlowInterval>
                    <Class1Count>6</Class1Count>
                    <Class2Count>8</Class2Count>
                    <Class3Count>9</Class3Count>
                    <FlowStatusTypeRef>1</FlowStatusTypeRef>
                </DetectorFlowDynamic>
                <DetectorHeadwayDynamic>
                    <LastUpdated>2001-12-17T09:30:47.0Z</LastUpdated>
                    <Headway>P1Y2M3DT10H30M</Headway>
                    <HeadwayInterval>P1Y2M3DT10H30M</HeadwayInterval>
                    <HeadwayStatusTypeRef>2</HeadwayStatusTypeRef>
                </DetectorHeadwayDynamic>
                <DetectorOccupancyDynamic>
                    <LastUpdated>2001-12-17T09:30:47.0Z</LastUpdated>
                    <Occupancy>3.14159E0</Occupancy>
                    <OccupancyInterval>P1Y2M3DT10H30M</OccupancyInterval>
                    <OccupancyStatusTypeRef>34</OccupancyStatusTypeRef>
                </DetectorOccupancyDynamic>
                <DetectorQueueDynamic>
                    <LastUpdated>2001-12-17T09:30:47.0Z</LastUpdated>
                    <QueuePresent>true</QueuePresent>
                    <QueueSeverityTypeRef>45</QueueSeverityTypeRef>
                </DetectorQueueDynamic>
                <DetectorSpeedDynamic>
                    <LastUpdated>2001-12-17T09:30:47.0Z</LastUpdated>
                    <Speed>3.14159E0</Speed>
                    <SpeedInterval>P1Y2M3DT10H30M</SpeedInterval>
                    <SpeedStatusTypeRef>56</SpeedStatusTypeRef>
                </DetectorSpeedDynamic>
            </Dynamics>
            <History>
                <DetectorHistory>
                    <LogId>21</LogId>
                    <LogDate>2009-12-23T09:30:47.0Z</LogDate>
                    <LogEntry>String</LogEntry>
                    <LogUser>String</LogUser>
                </DetectorHistory>
                <DetectorHistory>
                    <LogId>21</LogId>
                    <LogDate>2009-12-27T09:30:47.0Z</LogDate>
```

© Copyright Kizoom 2007-2009

```
                                        <LogEntry>String</LogEntry>
                                        <LogUser>String</LogUser>
                                </DetectorHistory>
                        </History>
                        <Faults>
                                <DetectorFault>
                                        <FaultTypeRef>1</FaultTypeRef>
                                        <CreationDate>2001-12-17T09:30:47.0Z</CreationDate>
                                        <ClearedDate>2001-12-17T09:30:47.0Z</ClearedDate>
                                        <Description>String</Description>
                                        <DataSourceTypeRef>2</DataSourceTypeRef>
                                        <FaultId>345</FaultId>
                                        <SupplierFaultNumber>String</SupplierFaultNumber>
                                        <SubSystemTypeRef>0</SubSystemTypeRef>
                                        <EquipmentFault>true</EquipmentFault>
                                        <CommunicationsFault>false</CommunicationsFault>
                                        <FaultSeverity>1</FaultSeverity>
                                        <AcknowledgmentDate>2001-12-
17T09:30:47.0Z</AcknowledgmentDate>
                                        <AcknowledgementStateTypeRef>2</AcknowledgementStateTypeRef>
                                </DetectorFault>
                        </Faults>
                        <Commands>
                                <DetectorCommand>
                                        <RequestTime>2001-12-17T09:30:47.0Z</RequestTime>
                                        <CommandId>1</CommandId>
                                        <Command>Shut up</Command>
                                        <CommandFormatTypeRef>0</CommandFormatTypeRef>
                                        <Priority>1</Priority>
                                        <DataSourceTypeRef>0</DataSourceTypeRef>
                                        <AcknowledgementTime>2001-12-
17T09:30:47.0Z</AcknowledgementTime>
                                        <ResponseTime>2001-12-17T09:30:47.0Z</ResponseTime>
                                        <ResponseResult>1</ResponseResult>
                                        <ResponseComment>String</ResponseComment>
                                </DetectorCommand>
                        </Commands>
                </Detector>             </DataObjectDelivery>
</ServiceDelivery>
```

DATA OBJECT SUBSCRIPTION REQUEST EXAMPLE

The following further example shows a ***DataObjectSubscriptionRequest*** to set up a subscription to be sent updates for a stop asynchronously. It embeds a ***DataObjectRequest*** which specifies the selection criteria **-** .

```
<siri:Siri xsi:schemaLocation="http://www.siri.org.uk/siri ../../utmc.xsd" xmlns:siri="http://www.siri.org.uk/siri"
xmlns="http://www.utmc.uk.com/utmc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <siri:SubscriptionRequest>
                <siri:RequestTimestamp>2004-12-17T09:30:47-05:00</siri:RequestTimestamp>
                <siri:RequestorRef>NADER</siri:RequestorRef>
                <DataObjectSubscriptionRequest>
                        <siri:SubscriberRef>NADER</siri:SubscriberRef>
                        <siri:SubscriptionIdentifier>Sub1234</siri:SubscriptionIdentifier>
                        <siri:InitialTerminationTime>2001-12-17T09:30:47.0Z</siri:InitialTerminationTime>
                        <SubscriptionFilters>
                                <TrafficEventSubscriptionFilter>
                                        <TrafficEventRequestFilter>
                                                <Topics>
                                                        <Sources>
                                                        <DataSourceTypeRef>2</DataSourceTypeRef>
                                                        </Sources>
                                                        <Current/>
                                                        <Types>
                                                                <AllIncidentTypes/>
                                                        </Types>
                                                </Topics>
                                                <Policies>
                                                        <IncludeDeleted>false</IncludeDeleted>
                                                <DataRequestType>elements</DataRequestType>
                                                </Policies>
                                        </TrafficEventRequestFilter>
```

© Copyright Kizoom 2007-2009

```
                                    </TrafficEventSubscriptionFilter>
                                </SubscriptionFilters>
                            </DataObjectSubscriptionRequest>
                        </siri:SubscriptionRequest>
```

SIRI DATA OBJECT SUBSCRIPTION DELIVERY EXAMPLE

The following is an example of a ***DataObjectDelivery*** in response to a ***DataObjectSubscription-Request***. It is almost identical to the earlier response to a ***DataObjectRequest,*** but also includes subscription identifier data.

```
<ServiceDelivery>
        <!--=======HEADER================================================== -->
        <ResponseTimestamp>2004-12-17T09:30:46-05:00</ResponseTimestamp>
        <ProducerRef>KUBRICK</ProducerRef>
        <!--=======PAYLOAD================================================ -->
        <DataObjectDelivery version="0.1d">
                <ResponseTimestamp>2004-12-17T09:30:47-05:00</ResponseTimestamp>
                <SubscriberRef>NADER</SubscriberRef>
                <SubscriptionRef>2004-12-17T09:30:47-05:00</SubscriptionRef>
                <ValidUntil>2004-12-17T09:30:47-05:00</ValidUntil>
                <!--====FIRST ARRIVAL ============================================== -->
        <DataObjects>
                <Incident>
                        <SystemCodeNumber>INc00001234</SystemCodeNumber>
                        <Elements>
                                <IncidentElement>
                                        <ObjectTypeRef>456</ObjectTypeRef>
                                        <QualityRef>54</QualityRef>
                                        <CreationDate>2009-04-01T09:30:47.0Z</CreationDate>
                                        <DataSourceTypeRef>56</DataSourceTypeRef>
                                        <Reality>real</Reality>
                                        <ShortDescription>Shagup on the M99</ShortDescription>
                                        <LongDescription>Shagup on the M99</LongDescription>
                                        <Point id="po1" >
                                                <Easting>999999</Easting>
                                                <Northing>1999999</Northing>
                                        </Point>
                                        <NetworkPathRef>a</NetworkPathRef>
                                <LocationDescription>Two milesalong the M99 past t J12</LocationDescription>
                                        <Name>Jam</Name>
                                        <ReportedBy>Mr Keen</ReportedBy>
                                        <ConfirmedBy>Mr Eager</ConfirmedBy>
                                        <ModifiedBy>Ms Right</ModifiedBy>
                                        <AffectedZoneRef>123</AffectedZoneRef>
                                        <ExternalReference>
                                                <CountryCode>uk</CountryCode>
                                                <SystemCode>HA45</SystemCode>
                                                <SituationIdentifier>HA12245</SituationIdentifier>
                                                <SituationVersion>001</SituationVersion>
                                        </ExternalReference>
                                        <IncidentTime>2009-04-01T09:30:47.0Z</IncidentTime>
                                </IncidentElement>
                        </Elements>
                        <Extensions/>
                </Incident>

                <Accident>
                        <SystemCodeNumber>Acc00001234</SystemCodeNumber>
                        <Elements>
                                <AccidentElement>
                                        <ObjectTypeRef>456</ObjectTypeRef>
                                        <QualityRef>23</QualityRef>
                                        <CreationDate>2009-04-01T09:30:47.0Z</CreationDate>
                                        <DataSourceTypeRef>54</DataSourceTypeRef>
                                        <Reality>real</Reality>
                                        <ShortDescription>Crash in Scunthorpe centre</ShortDescription>
                                        <LongDescription>Crash in Scunthorpe centre</LongDescription>
```

© Copyright Kizoom 2007-2009

```
                                        <Point id="NMTOKEN" >
                                                <Easting>999999</Easting>
                                                <Northing>1999999</Northing>
                                        </Point>
                                        <NetworkPathRef>a</NetworkPathRef>
                                        <LocationDescription>String</LocationDescription>
                                        <Name>normalizedString</Name>
                                        <ReportedBy>Mr Keen</ReportedBy>
                                        <ConfirmedBy>normalizedString</ConfirmedBy>
                                        <ModifiedBy>normalizedString</ModifiedBy>
                                        <AccidentTime>2009-04-01T09:30:47.0Z</AccidentTime>
                                <EndTime>2009-04-01T12:30:47.0Z</EndTime>
                                </AccidentElement>
                        </Elements>
                        <Extensions/>
                </Accident>
                <RoadWorks>
                        <SystemCodeNumber>RD34567</SystemCodeNumber>
                        <Elements>
                                <RoadWorksElement>
                                        <ObjectTypeRef>65</ObjectTypeRef>
                                        <QualityRef>35687</QualityRef>
                                        <CreationDate>2009-01-17T09:30:47.0Z</CreationDate>
                                        <DataSourceTypeRef>32</DataSourceTypeRef>
                                        <ShortDescription>a</ShortDescription>
                                        <LongDescription>a</LongDescription>
                                        <Point id="po22" >
                                                <Easting>999999</Easting>
                                                <Northing>1999999</Northing>
                                        </Point>
                                        <NetworkPathRef>a</NetworkPathRef>
                                        <TransportLinkRef>a</TransportLinkRef>
                                <LocationDescription>Ressrfacing at Hobbits roundabout</LocationDescription>
                                        <Name>normalizedString</Name>
                                        <ReportedBy>Gordon</ReportedBy>
                                        <DiversionRoute>String</DiversionRoute>
                                        <DiversionInForce>true</DiversionInForce>
                                        <ExternalReference>
                                                <SituationIdentifier>EJX234</SituationIdentifier>
                                        </ExternalReference>
                                        <Planned>
                                                <StartTime>2009-05-17T09:30:47.0Z</StartTime>
                                                <EndTime>2009-05-17T05:30:47.0Z</EndTime>
                                        </Planned>
                                        <Actual>
                                                <StartTime>2009-06-17T09:30:47.0Z</StartTime>
                                                <EndTime>2009-06-17T05:30:47.0Z</EndTime>
                                        </Actual>
                                        <Contractor>S.Hifty</Contractor>
                                        <TrafficSignals>true</TrafficSignals>
                                        <Contraflow>true</Contraflow>
                                        <ConfirmedDate>1967-08-13</ConfirmedDate>
                                </RoadWorksElement>
                        </Elements>
                </RoadWorks>



        </DataObjects>
        </DataObjectDelivery>
</ServiceDelivery>
```

© Copyright Kizoom 2007-2009

9. ANNEX – A SIRI COMMON DATA TYPES

COMMON SIRI DATA TYPES – XML SIMPLE TYPES

The SIRI-SX services use XML data types for primitive data types where possible (Figure 9-1).

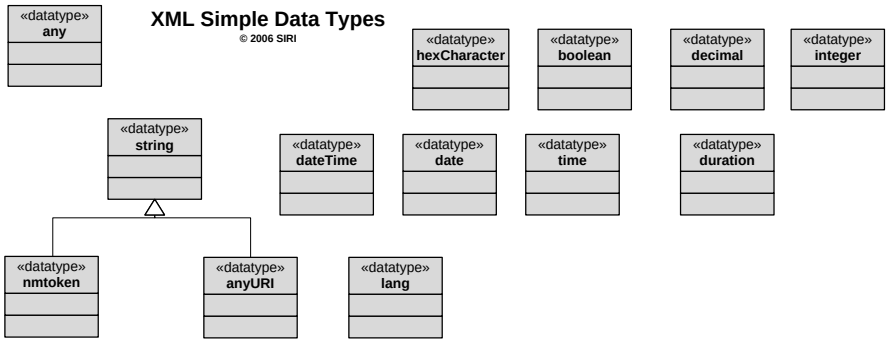


Figure 9-1 XML simple data types

COMMON UTMC DATA TYPES

The UTMC-SX services use a number of common UTMC data types (Figure 9-2).

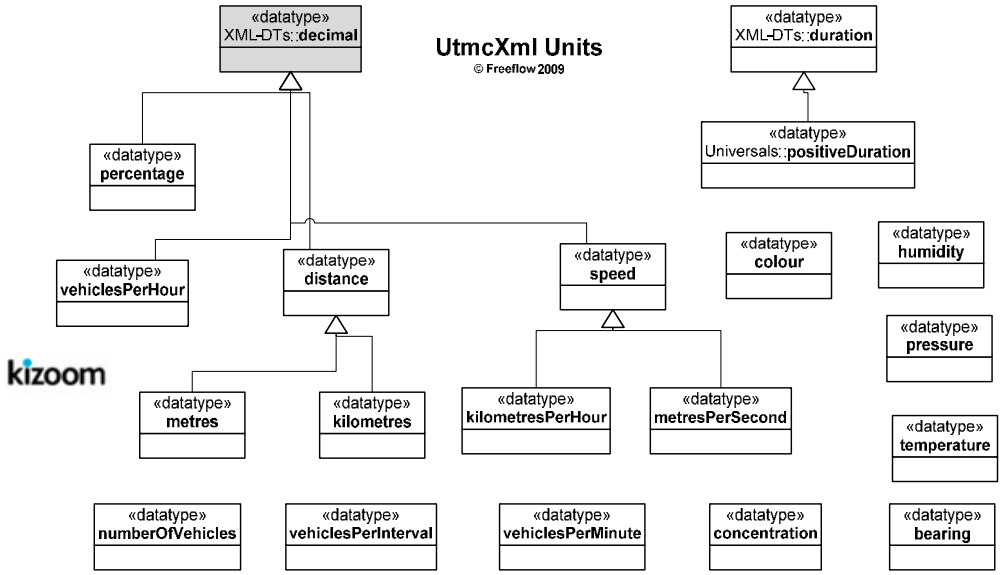


Figure 9-2 UML Diagram of Common SIRI Data Types

© Copyright Kizoom 2007-2009

UTMC DAY & DATE TYPES

UtmcXml use a common representation of day types

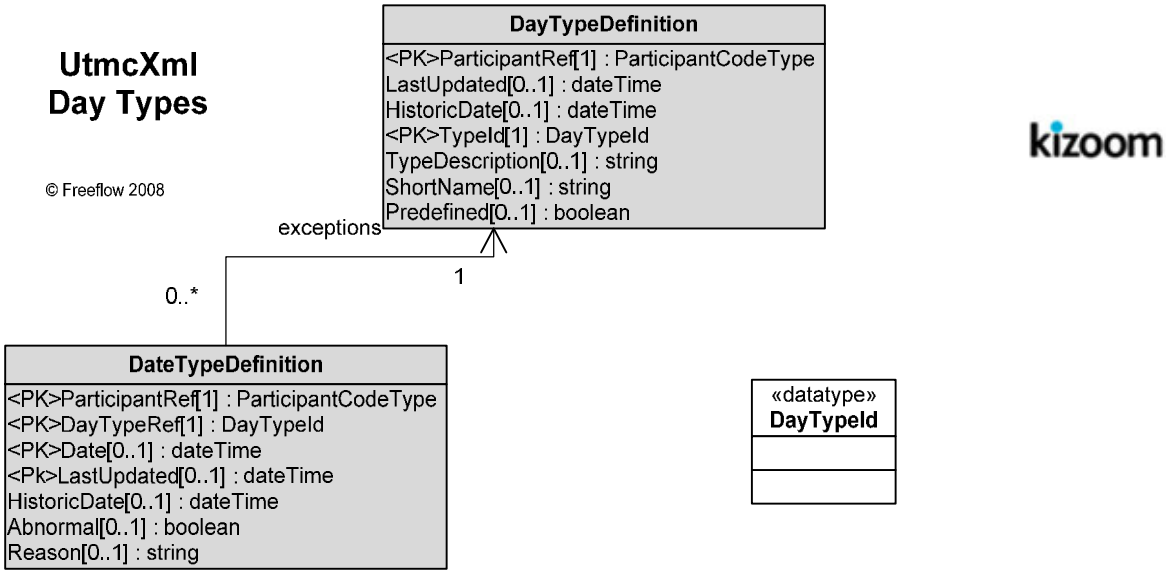
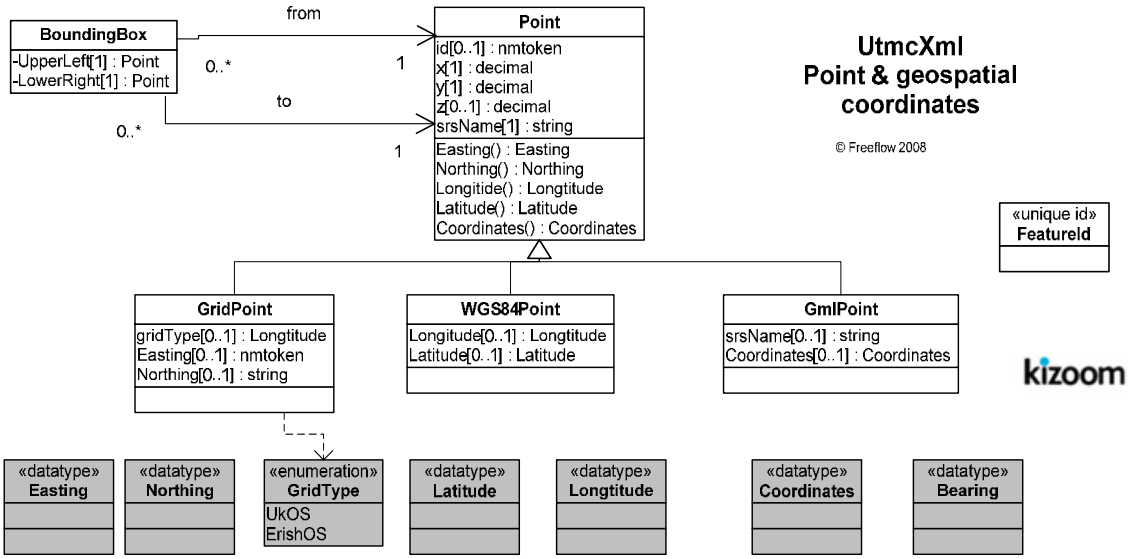


Figure 9-3 UTMC Day & Date Types

UTMC POINT

UtmcXml use a common representation of Point. Normally this will be expressed as OS Easting & Northing but other coordinates may be provided



© Copyright Kizoom 2007-2009

COMMON SIRI SIMPLE DATA TYPES – CODES & IDENTIFIERS

As well as the XML simple data types, SIRI uses a number of additional simple data types see Figure 9-4. Many of these are just explicitly typed identifiers for entities – These have primitive type of string or NMTOKEN).

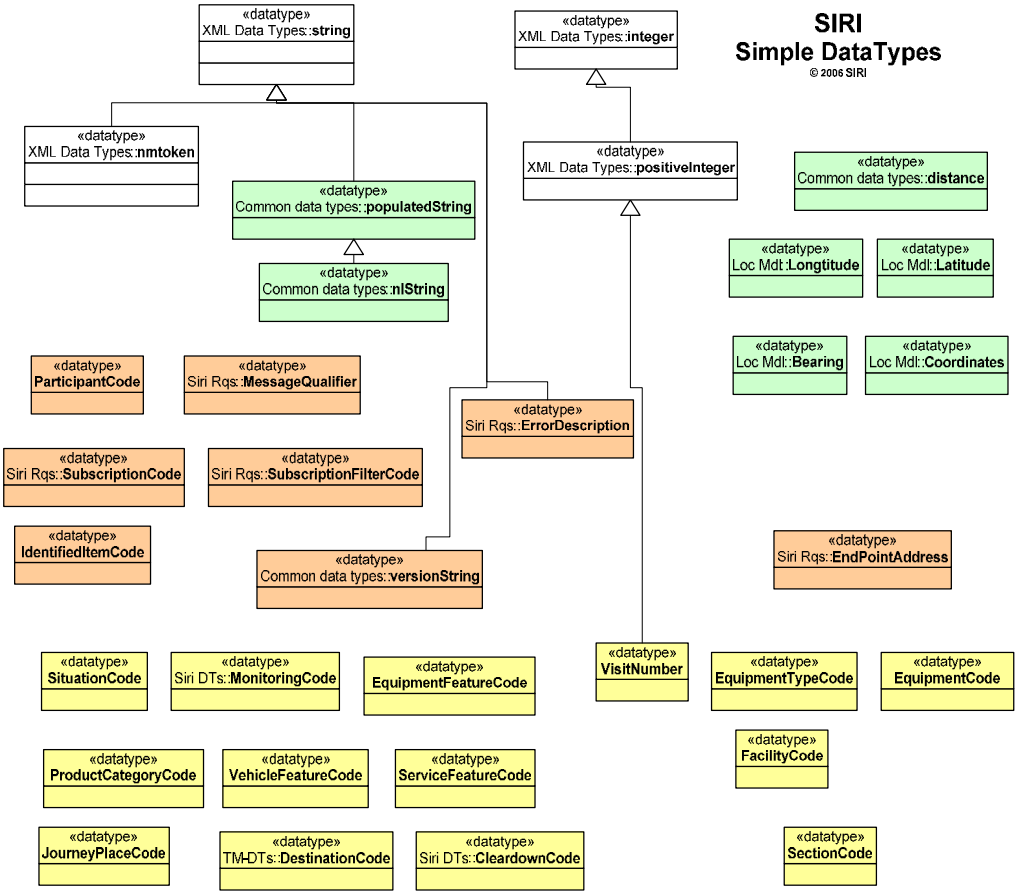


Figure 9-4 Common SIRI Simple Data types

© Copyright Kizoom 2007-2009