Cross-industry Remote Condition Monitoring and Data Sharing - Architecture Principles and Requirements

I RESEARCH AND DEVELOPMENT

Copyright

Written by: Pete Johnson, Interfleet Rail

Published: March 2015

Contents

Architecture principles and requirements

References and definitions

Abbreviations and glossary

Term	Definition
Advisory Generation	The sixth stage in the 6-level processing model for condition monitoring defined in ISO13374. Work orders, recommendations, and so on, are issued based on the prognostic assessment.
AIXM	Aeronautical Information Exchange Model – framework to enable management and distribution of Aeronautical Information Services data.
Algorithm	A method of calculating a result based on input data.
API	Application programming interface
Architecture	From TOGAF®: 1 A formal description of a system, or a detailed plan of the system at component level, to guide its implementation (source: ISO/IEC 42010:2007). 2 The structure of components, their inter-relationships, and the principles and guidelines governing their design and evolution over time.
ATOC	Association of Train Operating Companies
AVI	Automatic vehicle identification
BLOB	Binary large object. A chunk of binary data, usually representing multimedia or other encoded representation. Typically, the database where data are stored in this way does not know or care what the format of the data is.
CCS	Control and command systems
CRS	Centralised Reservations System. Original computerised reservations system now replaced, which gave its name to the 3-character location codes used to identify railway stations.

Term	Definition
CSV	Comma-separated values
Conceptual model	A data model showing how the core concepts of an area of interest are defined and are related to each other, independently of any particular way of storing or exchanging the data.
Data acquisition	The first stage in the 6-level processing model for condition monitoring defined in ISO13374. Data are captured from sensors.
Data manipulation	The second stage in the 6-level processing model for condition monitoring defined in ISO13374. Data from a sensor undergoes basic summarising cleansing operations such as averaging or signal processing.
Data model	A way of expressing the data items associated with an area of interest and the relationships between them.
Deterioration profile	A graph describing the expected rate of deterioration of some characteristic of an asset from an assumed level of usage or passage of time.
DRACAS	Defect Resolution and Corrective Actions System
EIA	Enterprise integration architectures – an approach to integrating IT systems to support cross organisational business processes.
ELR	Engineers Line Reference – a designator for a section of rail route between junctions.
Encapsulation	The property of a system whereby only the elements of any object or data item that need to be made accessible to the outside world are visible; all other elements are kept private. Systems exhibiting this characteristic are easier to extend and develop.
Enterprise Architect	Data modelling tool used in T1010-01 to model the various artefacts of the project (such as sources, recommendations, dependencies, use cases) and traceability between them.
ESB	Enterprise service bus – a component of Enterprise Integration architectures whereby all participating IT systems interact via a single 'bus' with standard data formats, rather than directly with each other.

Term	Definition
ETL	Extract, transformation, load. The component of a data warehousing system responsible for capturing, cleaning, organising and storing the data to be reported or analysed.
EVN	European Vehicle Number
Extensibility	The property of a system which enables its capability to be enhanced easily after initial construction by adding new functions without disrupting existing ones or their users.
HABD	Hot Axle Box Detection System
HMAC	Hash-based message authentication code. A way of signing datagrams. Defined in RFC 2014 [1]
Health Assessment	The fourth stage in the 6-level processing model for condition monitoring defined in ISO13374. The health of the asset or component asset is assessed using the state from the previous stage and any supplementary information.
INSPIRE	Infrastructure for Spatial Information in the European Community – European Directive aimed at creation of data infrastructure for spatial data.
InteGRail	European research project into information integration and management across the rail industry
Intelligent Infrastructure	Network Rail strategy to deliver infrastructure management improvements through intelligence design and maintenance through prediction and prevention rather than fixing.
JSON	Javascript Object Notation. A simple text-based way of representing structured information.
LINX TM	Network Rail initiative to enable open interchange of information about the control of moving trains using EIA components to handle interfaces between traffic management and other industry systems. See EIA.
Logger	A device which records, stores and passes on the data from one or more sensors.
LUL	London Underground Ltd.

Term	Definition
Metadata	'Data about data'. Information about the format and type of data; possibly also including information about how it can be used.
MIME Type	File format recognised by the Internet Assigned Numbers Authority as a standard format in its list of media types [2].
MIMOSA	Operations and Maintenance Information Open System Alliance – association for development of open information standards for operations and maintenance in manufacturing, fleet, and facility. Enables collaborative asset lifecycle management.
Modularity	The property of a system in which the functions and data are organised in groupings (modules) by purpose, with a small number of clear relationships between the modules. This property makes the system more extensible and maintainable because it limits the spread of impact of any single change.
NALCO	National Location Code
NETEX	Network Exchange – XML schema based standardisation initiative for data formats and interchange methods for railway data, used widely in Europe.
NR	Network Rail
OAUTH	
OLE	Overhead line equipment
Ontology	A schema where objects are defined in relation to other objects to allow understanding of a new concept by a machine.
Ontology Engineering	A set of processes associated with constructing and maintaining an ontology.
ORBIS	Offering Rail Better Information Services – a Network Rail programme to improve its approach to acquisition, storage and usage of asset information
OSI	Open Systems Interconnect. A layered architecture for organising the networking together of computers.

Term	Definition
OWL	Web ontology language – a language used to create ontologies. OWL is recommended by the Internet's governing body World Wide Web Consortium (W3C).
Prognostic Assessment	The fifth stage in the 6-level processing model for condition monitoring defined in ISO13374. An estimate of the future health of the asset is made using the health assessment and any supplementary information about environment.
R2	An IT initiative intended to replace the Rolling Stock Library and the Rail Vehicle Record System (RAVERS)
railML	Railway Markup Language - data standard for the interchange of railway industry, in the form of XML schemas. See XML.
RCM	Remote condition monitoring –the activity of monitoring the condition of assets remotely: via a sensor at or on the asset which sends data to another location.
REST	Representational state transfer. A style of web application architecture in which the history and state of a user's interaction is shown in the web pages themselves rather than being remembered by the server.
RFA	Railway functional architecture
RFC	Request for comment. An internet standard maintained by the Internet Engineering Task Force.
RFID	Radio frequency identification – a system whereby tags wirelessly transmit identification data to a sensor.
RIS	Rail Industry Standard - a definition of technical or functional requirements. In this document it is the Traction and Rolling Stock RIS which is relevant.
Schema	A formal definition of a set of data structures. A specific type of schema that will be used is an XML schema (http://www.w3.org/XML/Schema.html) which enables the structure and valid content of an XML file to be defined formally in a way that can be checked automatically in software.

Term	Definition
Semantic	Pertaining to the meaning of data: what it represents and how that affects its relationship with other data.
Semantic metadata	Metadata which describes a way of defining the meaning of data. An ontology is a type of semantic metadata.
Sensor	A device which converts a physical property of the environment or of an asset into an electrical signal.
Server	A computer which hosts a software service which can be used by other connected computers.
SQL	Structured query language. Standard language for querying relational databases.
State Detection	The third stage in the 6-level processing model for condition monitoring defined in ISO13374. Processed data from sensors is assessed in relation to a threshold or band of acceptability and alerts or alarms raised if exceeded.
TIPLOC	Timing Point Location code
TLS	Transport Layer Security – a web encryption protocol defined in RFC 5246 [3].
TLS – OB and MT	Train Location Strategy – RSSB project to enable provision of train location information to application providers to an industry standards through collation, consolidation and distribution of information from trains and signalling. TLS OB refers to on-board systems which send location information to the central system. TLS MT (multi-train). MT will enhance the accuracy of this information based on data from many trains and send it back to the train OB system.
TOGAF	The Open Group Architecture Framework (http://www.opengroup.org/togaf/) A standard methodology for defining software architectures.
TOPS	Total Operations System
TRUST	Train Running System on TOPS

Term	Definition
TSI	Technical Specifications for Interoperability – European Standards for the interoperability of data including TAP (Telematics Applications for Passenger Services) and TAF (Telematics Applications for Freight)
TSV	Tab-separated values
UIC	Union Internationale des Chemins de Fer – International Railways Union
UJG	Uninterrupted Journey Group
UOMS	Unattended Overhead Line Management System
UUID	Universal unique identifier. A standard type of identifier generated by a computer algorithm, (nearly) guaranteed to be completely unique[a]. Can be represented by a 128-bit binary number or a 36-character string of hexadecimal digits with spacers.
VPN	Virtual Private Network
XiRCM	Cross-industry remote condition monitoring (in the rail industry) – remote condition monitoring occurring in: 1 any of the quadrants where train based sensors monitor train, or infrastructure, and infrastructure based sensors monitor train, or infrastructure 2 across different parties and sections of the industry – different groups may be involved in the operation of an asset, provision of a sensor, monitoring of that sensor.
XiRCM Programme Phase 2	Cross-industry Remote Condition Monitoring Programme Phase 2 – a research project to facilitate the introduction of more cross industry remote condition monitoring, comprising packages to deliver architecture (01), commercial recommendations (02), review of standards (03) and decision support tool extension (04).
XiRCMSG	Cross Industry Remote Condition Monitoring Strategy Group – a subgroup of the Vehicle/Vehicle System Interface Committee with the responsibility to deliver the XiRCM Programme.

Term	Definition
XML	Extensible Markup Language – a widely used language used to define data structures (see Schema) whereby elements and attributes and their structure can be declared and assigned values.
XSLT	Extensible Stylesheet Language - Transformations

a. If 1,000,000,000 UUIDs were generated every second for the next 100 years, there is a 50 % chance that there would be 1 duplicate [28].

Expressions of requirements

In the core of this document, requirements on the RCM data architecture, its managers or its users are shown **numbered 00nn and in bold text**. In the text of the requirements, the following key words are to be interpreted as described in RFC 2119 [4]: 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL'.

In some cases, additional key words may be used based on the extended set defined in RFC 6919 [5].

Introduction

The project

In 2013 the Cross-industry Remote Condition Monitoring Strategy Group, a subgroup of the Vehicle/Vehicle System Interface Committee, launched the Cross-industry Remote Condition Monitoring Programme Phase 2. The purpose of the programme is to support the industry's high-level objectives of improving reliability, capacity and value for money, detailed in reports such as the Rail Technical Strategy 2012: The Future Railway and Rail Value for Money 2011.

The RSSB-managed research programme T1010 is intended to set up the enabling framework for this initiative. T1010 is in 4 parts:

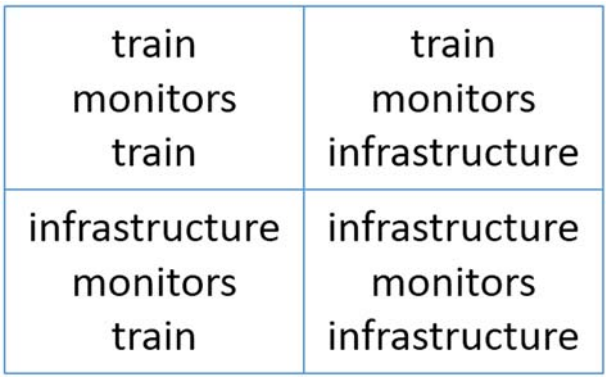
- T1010-01 – the current project – is to define a data architecture to support cross-industry RCM data sharing
- T1010-02 is to define a commercial and legal framework to be used by parties wishing to share RCM data
- T1010-03 will update industry standards and guidance to promote and simplify the use of the data sharing architecture
- T1010-04 will update the business case assessment tool for RCM data sharing initiatives.

In 2013 Halcrow (now known as CH2M HILL) was awarded the contract T1010-01 to develop the cross-industry remote condition monitoring data architecture for the UK rail industry. The data architecture is intended to remove some of the technical barriers to cross-industry data sharing caused by incompatibilities of approach between stand-alone RCM projects. It will do this by defining standard methods for structuring, representing and interchanging RCM data.

Remote condition monitoring is an aspect of asset management where the condition of an asset is monitored using data output from a sensor, so it can if necessary be monitored remotely from the asset itself (a person or system in a base station receives data directly from an asset on the railway).

Railway remote condition monitoring activities can be partitioned into 4 quadrants, defined by which types of asset – trains or infrastructure- are being monitored; and where – on trains or on infrastructure – the monitoring is being done. In this view, the 2 cross-interface quadrants - trains monitoring infrastructure and infrastructure monitoring trains - are of particular focus in this research as they involve different industry parties.

Figure 1 - The remote condition monitoring quadrants



However, the 'cross-industry' element of RCM data sharing is not limited to this view. Sharing of RCM data on the same side of the rail boundary can deliver benefits by breaking down information silos. One of the visions for the rail industry discussed in the Rail Technical Strategy 2012 is the concept of a whole system approach, with aligned asset management practices. The greater availability of shared data through the industry will enable a greater level of condition monitoring to take place with resulting improvements to reliability. This is in contrast to the current situation where many RCM systems in use and being developed operate as stand-alone systems where the data types, communication protocols and architecture do not necessarily conform to any agreed standards and can be tied to the equipment vendor.

Context

Work done so far on T1010-01 has produced a *Review of relevant RCM developments* [6], which has: surveyed previous work on data integration in the rail industry; on rail RCM data integration in the UK and elsewhere; on relevant developments in the IT industry; on RCM data integration initiatives in other industries; and on the legal and contractual background to rail RCM activities.

Also in that document is a review of consultation with relevant parties: current RCM activity sponsors, academics, RCM system suppliers and the people responsible for new IT and information services in rail asset management.

The results of that work were codified into a set of Influences on the RCM data architecture which inform its future shape and design. This document builds on those to define a set of RCM Data Architectural Principles and architecture Requirements which should inform the development of an industry-wide RCM data architecture and guide those responsible for implementing and managing it and participating in it. A repository of all the requirements and the relationships between them has been created and accompanies these documents. The repository is held in a Sparx Enterprise Architect file.

Purpose of the architecture

The role of the data architecture is to provide an easily-adopted and stable way for RCM data to be interchanged across the UK rail industry. To make the architecture stable, it will be based on open and common standards; to make it easily-adopted, it will be closely mapped to the types of data interchange used and foreseen for the UK industry.

Scope of the architecture

RCM data can be shared between parties in order to support business processes which have goals such as reliability improvement, maintenance of safety and cost reduction. These processes also require data from other sources such as: asset management systems, operational systems, and work planning systems. They are informed by RCM data, but also involve data of their own which are not directly RCM-related.

The RCM data architecture described in this document covers the following scope:

- Data directly associated with remote condition monitoring: sensor data in raw and processed forms; alerts and alarms; health assessments and estimates of usable life or future condition.
- Data needed to identify the assets described by or affected by the RCM data
- External data on usage or other influences on future life or state such as weather

The architecture does not restrict the type of rail asset being considered (moving or fixed, point or linear), the type of data being collected (analogue

or digital, time-based, frequency-based or event-driven, alerts, alarms and health assessments), or the level of analysis (raw, processed, assessed against tolerance criteria, merged with other data). It will therefore potentially work with any current rail RCM initiative and with any reasonable ones in the future.

The architecture does not demand large-scale investment in hardware or software, though some of the more sophisticated applications at the higher levels will benefit greatly from some shared IT infrastructure, for example to manage shared reference data and to mediate message transfers between data originators and data consumers.

Use of the architecture

It is envisaged that the architecture principles and architecture requirements will have these uses:

- As a set of specific requirements on data sharers which can be included in agreements between data providers and data users, which will simplify their own task of interchanging data while furthering the broader goals of industry data availability
- As requirements on the developers of any shared infrastructure which is set up to support data interchange in an industry-standard way
- As guidance on the principles to be followed in managing the architecture over the longer period to ensure that its goals are realised

This document

The first part of the document considers the overall style of the proposed RCM data architecture and sets out the underlying principles it should adopt.

- Section 3 gives an overview of software architectural styles
- Section 4 analyses the fit of the different styles against the goals and influences identified in the *Review of relevant RCM developments* and concludes on a recommended style for the RCM data architecture
- Section 5 outlines the proposed RCM data architecture structure in terms of technical layers, levels of evolution and compliance and the business process support that each compliance level offers

The following sections describe the elements of the architecture in more detail:

- Section 6 describes the data models required to support the sharing of RCM data in the rail industry: a shared understanding of what data entities need to be recognised and the relationships between them
- Section 7 describes the shared reference data which will be needed to support data sharing using the data architecture
- Section 9 defines some standards for representing metadata which will help data users understand limitations of the shared RCM data in terms of ownership, data quality, licensing and usage rights
- Section 10 defines standard ways in which data can be requested and provided using the data architecture
- Section 11 covers data integrity and security concerns
- Section 12 identifies shared reference data services which will need to be provided by 3rd parties
- Section 13 considers the tasks necessary to set the data architecture up and manage it

In these sections, specific requirements for the architecture and its managers have been highlighted and are summarised in Appendix 4.

Other appendices to the document are:

- Appendix 1 – Sample XML schemas
- Appendix 2 – Sample datagrams built using the schemas
- Appendix 3 – works referenced in the text

The appendices are published as separate documents.

Software architectural styles – an overview

Introduction

Several different styles of software architecture have evolved and are being developed to meet different types of business requirement. Data architecture forms an important part of the software architecture. In this section we give an overview of the main architectural styles and explain the pros and cons of each style in terms of the RCM data architecture requirements.

Following this, we describe the approach chosen for the RCM data architecture and outline its key characteristics and themes. These are introduced and stated as requirements or recommendations in this document; detail and examples will be added in the next phase of the work.

Software architecture styles

The following sections describe some common and developing architecture styles and patterns. The information is gathered from several sources, but particularly from Fielding's 2000 paper [7] which introduced the REST concept on which many recent software architectures have been based.

Layered architectures

A fundamental concept applied in many software architectures is that of layering. The overall task of the architecture is subdivided into elements which each carry out a single clear function and which only communicate with their immediate neighbours via a clearly-defined interfacing protocol. The purpose is to clarify the design of software which implements the architecture so that it is robust and maintainable. It helps to embody desirable design principles:

- Encapsulation: users of a service don't need to see inside it or know how it works to use it
- Modularity or separation of concerns: conceptually-linked functions are in distinct elements of the architecture so that they can be maintained and

upgraded separately from each other without affecting the system's operation

- Loose coupling and high cohesion: each module only interacts with a small number of others and in a simple way; and contains all the functionality related to its purpose

A classic example of this type of approach is the Open Systems Interconnection (OSI) model which describes, among other things, how Internet communication works. This is a 7-layer model, with layers shown in Table 1.

Table 1 - OSI model - example of layering

Layer	Example	Note
7 Application	HTTP - Hypertext Transfer Protocol, used by web servers and browsers	A browser and a web server can communicate with each other using HTTP without having any idea how the connection between them is maintained.
6 Presentation	The encryption layer of the HTTPS secure HTTP protocol	This layer encrypts the http traffic without having any idea what it contains or its purpose.
5 Session	Some aspects of TCP or Remote Procedure Calls	Starting, managing and terminating communication sessions between devices.
4 Transport	TCP – Transmission Control Protocol, used to route data packets reliably between computers.	Enables reliable connections between computers. Includes features for retrying, re-connecting, re-sending lost data packets etc. Does not care about the content of the data being sent; does not know how the data are physically routed.
3 Network	IP – Internet Protocol, used to route data between computers	Enables data, in the form of datagrams, to be transferred between devices identified by their IP address, taking care of any different routings possible. May not be reliable.

Table 1 - OSI model - example of layering

Layer	Example	Note
2 Data Link	PPP – point to point protocol, used to connect computers over telephone lines	Enables reliable communications to be passed between physical devices. Will manage data flow control, retry of failed packets, and others.
1 Physical	Ethernet, one of several protocols used to network computers together	Defines the electrical and physical characteristics of a method for computers to pass data between each other on the same network without knowing or caring what the data are. May not be reliable.

In this model, each layer has a distinct task to do – to communicate with the same layer in another program or on a different computer. It does this by communicating with the layer directly below it.

Client-server architecture

In this style of architecture, data and aspects of the processing are centralised in a server, while users and the presentation aspects of the system are distributed to client computers and connected via a network. Variations on the style have different amounts of the business logic at the server and the client.

This is a layered architecture, typically separated into a data layer, one or more business layers and a presentation layer.

The functions of the layers (sometimes known as tiers) are described in Table 2.

Table 2 - Client-server tiers

Layer	Typical function	Notes
Presentation layer	Interaction with the user	Client – on the user's computer. May be a web application or an installed program. Has a user interface to present data and gather data and instructions from the user

Table 2 - Client-server tiers

Layer	Typical function	Notes
Business layer (client)	Validation of data; some business logic	In a 'fat client' application, there may be quite a lot of business logic on the client; in a 'thin client' one, this would be restricted to, for example, data validation and format conversion.
Networking layer	Manage communication between client and server	The characteristics of the networking layer tend to have a heavy bearing on whether a 'fat client' or 'thin client' approach is used.
Business layer (server)	Business logic; management of transactions	Manages the interaction with the data storage medium in the data layer, for example by ensuring that only valid data from permitted users are allowed to be entered or altered. May handle complex validation or transactions in which several changes have to be done as a whole.
Data layer	Storage of persistent data	Manages the safe storage of data. Typically implemented as a relation database or similar. Will be designed to handle all the simultaneous accesses being made by system users with acceptable performance, while maintaining the integrity of the data

A classic example of this style is an application running on the World Wide Web, with a web server acting as the server, and users' browsers acting as clients.

In terms of RCM data, the architecture of the Intelligent Infrastructure programme is a client-server one, with sensors feeding data into a central server, to which users connect to extract alerts and alarms or carry out analysis and reporting.

Peer-to-peer architecture

In this architectural style, computers are connected to a network and each computer has a similar role to the others in terms of the services it offers and the needs it has. It is decentralised, as opposed to the centralised client-server model.

Characteristics of this type of architecture are:

- It has a 'self-discovery' function, where each connected computer can offer its services to the network and broadcast requests for information about services offered by others.
- It is resilient, because it can cope with the removal of individual computers or network interconnections.

Classic examples of this type of architecture are the file-sharing networks such as Bittorrent, Napster or Gnutella. In terms of remote condition monitoring, this architecture is used by networks of cheap sensors such as ZigBee[1].

Bus-based architecture

The concept of a 'bus' originally comes from electrical engineering, where a heavy-duty conductor runs through a piece of switchgear or other equipment and can be connected to at any point to provide current there. The term was then used in computer hardware to refer to the interconnections between areas of the computer typically implemented as a backplane. In this context, an example is the set of connections that runs along the motherboard of a PC which enables cards to be plugged in at any point. In computing networks, bus-style networks are distinguished from point-to-point ones. Ethernet is a bus-style network: a computer can connect at any point to the network and then communicate with every other computer on it.

In software terms, the concept implies an interconnection method where any software element connected to the bus can communicate with any other one. It only has one connection to the network, rather than individual connections to the other software elements. The rationale is that where many systems have to be connected together, the number of interfaces that need to be managed grows linearly with the number of systems; if systems were connected point-to-point, the number of interfaces grows quadratically and so much faster.

Implicit in the definition of a software bus is the notion of a common data format used by all communications using the bus. Any software module using the bus must translate its own data into the format used by the bus before it can transmit it; and must translate data it receives from the bus format to its own format. The translation tools are known as adaptors.

1 https://www.zigbee.org/Specifications.aspx

The most common software bus approach used in connecting separate computer applications together is the Enterprise Service Bus[2]. This combines the concepts of a bus data standard and adaptors with other technologies for passing messages between computers, for managing the distribution and delivery of messages and associated content, and for handling security and resilience.

In the rail industry, Network Rail has set up an enterprise service bus (ESB) architecture. One application using the approach is LINX TM, which is used to communicate traffic management messages.

Bus architectures are favoured where many different existing computer systems need to be connected together to support broader business processes; or where future expansion is a clear goal. They show their value when they enable a large number of point-to-point interfaces between systems to be replaced, since each collaborating system only needs to construct a single interface between itself and the ESB.

A potential role for the use of Web Ontologies (as described in the *Review of relevant RCM developments* [6] which precedes this one) is in the design and operation of adaptors. Ontological techniques are helpful for the mapping between different conceptual views of railway data or for shifting up or down in levels of detail (such as from vehicle to component level): such re-interpretations of data are likely to be necessary if adaptors are to be used.

Pipeline architecture

This architectural style refers to a set of cooperating software components, where each one reads a stream or file of incoming data, transforms it in some way and generates output in the form of another stream or file to be made available to the next component in line.

The architecture is typically used in the data processing steps of a data warehousing system, usually described by the acronym ETL, for extract, transform, load. In this, the extract phase manages the transmission of data from external sources; the transform phase carries out the sometimes complex tasks of integration of the data, cleansing of inaccurate or missing values, adding descriptions to codes by lookup, restructuring to suit analysis or reporting; and the load phase manages the population of the data warehouse tables.

2 http://www.oracle.com/technetwork/articles/soa/ind-soa-esb-1967705.html

Data processing pipelines are common within RCM systems. They are typically associated with those areas where voluminous raw data need to be processed to generate a summary signal which can be compared with benchmarks to generate alerts and alarms – for example acoustic bearing monitoring systems in which raw sound data from an array of microphones are combined and analysed over a number of stages to produce diagnostic output.

Service-oriented architecture

This style of architecture has been described in the *Review of relevant RCM developments* report [6]. It sees software functionality being built up by the means of separate software modules, each of which can offer a specific service to any other authorised to request from it. Typically, web services are used to manage the requests, operations and responses.

This style supports cross-functional business processes in which several different departmental computer systems may need to collaborate to facilitate the process. By packaging the functions of the departmental systems as services and connecting them using methods such as an Enterprise Service Bus, new software functions can be assembled from existing systems without requiring extensive re-building or the disruption of existing processes.

New software systems emerging in the UK Railway are tending to use this approach, particularly as a means to making good-quality reference data and up-to-date operational data updates available widely to data consumers in an efficient standards-compliant way. Examples are LINX-TM, ORBIS and R2, all of which are conceived or are being built to offer their functionality in the form of services.

Independent IT systems – file interchange

This represents the very lowest level of integration of IT systems and describes the state of affairs before formal data integration initiatives are followed.

Data may be interchanged between the users of different systems, but it is in an ad-hoc manner mediated by files or manual transcription of data.

How requirements affect architectural style

These styles of software architecture have evolved in response to the characteristics of the users, technological drivers and system requirements. Some typical questions which drive the architectural style are listed in Table 3.

Table 3 - Questions influencing architectural style

Question	Influence on architecture
Where are the users?	User communities may be centralised or distributed. They may need direct access to their own applications; or may be able to use shared, web applications. The distribution of processing power, applications and data may depend on the speed and quality of communications that are required between them and the users.
What types of business process are being supported?	Processes may be strictly-controlled with a predictable workflow, or more loosely-structured requiring different inputs and discussions each time. They may have the scope of an office, a department or function, an organisation or a collaborating group of organisations. The more ad-hoc the process, the more collaborators it involves and the more interactions and dependencies it requires, the more it tends to favour an asynchronous message-based architecture rather than one with centralised processing and/or data storage.
What existing software systems are involved?	Organisation-wide and multi-organisation processes typically require interaction with other computer systems which already support elements of the process. These will probably not have been built with interoperation with others in mind. The more that this is so, the more a bus-style architecture is favoured over one that involves direct system-to-system data interchange. Also, the more that the systems involve different conceptual views of the same data, the more benefit there will be from supporting an ontological layer which will enable the semantics of the data to be captured.
What performance considerations are there?	The business processes associated with RCM data range from near real-time (fault detection, evaluation of impact on railway, rectification), through daily (update of asset registers and statuses, recording of maintenance work done.), to long-term (analysis of historic data for trends; forecasting of asset time-to-fail). Each of these processes has its own requirements in terms of the timeliness of data and processing. The more stringent the performance demands, the more this tends to suggest architectures which centralise data or require high-bandwidth, high-reliability interconnections.

Table 3 - Questions influencing architectural style

Question	Influence on architecture
What controls are required?	Controls typically relate to user access, data security and data integrity. The more concern there is about the impact of unauthorised access to systems or data on safety, operational performance or commercial confidentiality, the more the architecture will tend to a closed form with higher barriers to entry and use.
Where are data at rest?	Data used by processes and systems can be stored in a variety of formats and locations. These can be SQL databases, local databases, spreadsheets, unstructured text in electronic documents, paper documents; and they can be local or remote to users. In the RCM context, raw data and various levels of processed data may be at a central server, at field sites or on trains. In the different data stores, data may be stored in different formats with different codings and standards; and with different assumed meanings for the same piece of data. The more that data are distributed, the more that data integration approaches such as Data Warehouse ETL (extract, transformation, load) or distributed bus-style architectures become preferred.
Who owns the data?	Data ownership can tend to lead to restriction in access and usage. Architectures which centralise and regulate access to data tend to result; as do those that support restrictions in readability and use of the data.

Table 3 - Questions influencing architectural style

Question	Influence on architecture
What data volume considerations are there?	RCM data start out potentially very voluminous, especially if representing a continuous capture process. Each processing step tends to reduce the volume by significant factors, but retrospective analysis tends still to need the data at the most detailed level available. Storage of the raw data and access to it are significant concerns. There are costs and technical challenges associated with storing and securing data in bulk; and the benefit of keeping the data may not accrue to the storer. Data may need to be moved and stored in centralised locations if it is required by a large number of users or very rapidly on request. This favours a centralised form of architecture.
Are there many different sources or destinations for data?	If more than a small number of business processes need to merge together data from several different sources, a distributed bus-based architecture would be suggested over a centralised one.
What data interchange bandwidth questions are there?	If business processes require large volumes of data to be accessed quickly, this places a bandwidth requirement on the interconnection method. This requirement can be met by using a distributed architecture with high-performance interconnections or by positioning data locally to the users so slow networking is avoided.

Table 3 - Questions influencing architectural style

Question	Influence on architecture
What types of data are being handled?	RCM data are captured in a wide variety of formats, some standard and some not; though the underlying data types are usually of some standard type such as multimedia (audio/video), time series varying quantity, and binary state transitions. Reference data can be made available in rail-industry standard file formats (such as timetables in .CIF format or one of the standard railML or SIRI formats). Many existing RCM systems use proprietary or closed formats to store and transfer their data. These may have been created to maintain control over the data or to optimise the use of bandwidth. The architecture may choose to keep data in these formats and just support the sharing of the files; or it may take a bus approach which tends to favour translation of data into a standard form for transmission and rendering.
Where are the applications and processes that work on the data?	Data may be needed by applications local to the user, by organisation-wide application suites or by shared industry applications. These applications may themselves produce data which are needed by other processes elsewhere. If the processing is mostly centralised and under the control of a single party, the architecture would tend towards a centralised API-based one. If the processing is distributed and many parties can contribute to it, the architecture will tend to a services-based one where any given processing capability can be provided by any permitted party.

Current RCM system architectures and data interchanges

In considering the most useful architecture to define for cross-industry RCM data integration, it is important to factor in the types of data likely to be handled and the types of interchange likely to be needed to improve the operation of current business processes and support the implementation of new ones.

The RSSB research report T853 described the architecture of RCM systems in use in the UK railway in 2010 and mapped their functions against the 6-level RCM process model established in ISO 13374 (and described in the *Review of*

relevant RCM developments report [6]). Their rating is largely still correct, though it does not include recent systems such as RailBAM which generates alerts and alarms as well as optionally providing history for specific assets (and so would rate at the Health Assessment level), or the nationwide expansion of Intelligent Infrastructure.

It is clear that there is no standardisation in the formats of data interchanged. Below, we summarise some of the systems and the types of data from them. This is based on the brief analysis shown in Appendix F Systems and data in the *Review of relevant RCM developments report* [6].

Alerts and alarms (ISO 13374 'State Detection')

Most of the current RCM systems are capable of issuing alerts (representing non-critical events) and alarms (representing critical ones requiring immediate attention). There is no standard format for an alert/alarm, no standard method for identifying the assets involved, and no standard mechanism for circulating them. Systems tend to release text messages or emails when alerts or alarms are generated; and/or they have web pages on which current alerts or alarms can be viewed.

Some of the systems (for example Intelligent Infrastructure, RailBAM, InteRRIS) have their own data warehouses which enable historic data for relevant assets to be queried and trends identified.

Asset Status (ISO 13374 'Health Assessment')

Few current RCM systems provide data at this level. The only ones that could are those with the capability of merging data about different sensors on the same asset, or of multiple readings on the same asset over time; and the logic to derive an overall health assessment for the asset from these readings. Only RailBAM of the systems observed can do this.

T853 indicates that enabling this level of RCM analysis will be necessary to achieve the reliability improvement goals envisaged for Cross Industry RCM.

Prognostics and work scheduling (ISO 13374 'Prognostic Assessment' and 'Advisory Generation')

No existing railway RCM systems can support these levels of system maturity. They represent longer-term goals of a cross-industry RCM initiative. The characteristic of these levels is that they overlap strongly with asset management processes and need large amounts of contextual information

about the railway network and the impact of various types of problem on its capability, about the costs of repair and renewal, and about the capacity and availability of resources to do maintenance and replacement.

Architectural styles and the data integration requirements

In the *Review of relevant RCM developments* [6], a set of architectural guiding principles was established from an analysis of previous work in this area and recent developments in the IT field. In this section we assess the key architectural styles described above against these requirements to determine which style fits them the best, bearing in mind the considerations set out in Section 3.

Figure 2 - Requirements and architectural styles

	Interaction Style			Data Management Style			Processing Style			
Architectural Requirement	Stand-alone	Client Server	Bus-based	Silo	Central Database	Federated	Silo	App Layer Centralised	App Layer Distributed	Service-based
Modularity	2	1	3	2	1	3	1	1	2	3
Extensibility	2	2	3	2	1	2	1	2	2	3
Interoperability	2	1	3	1	2	3	1	2	2	3
Reuse / alignment	2	2	2	1	2	2	1	2	2	3
Standards	1	1	3	1	3	3	1	1	1	2
Use of COTS	1	1	3	1	1	2	1	1	2	2
Service Oriented architecture	1	1	3	1	2	3	1	1	1	3
Incorporation of current data models	1	2	2	3	1	3	3	2	2	2
Agreed suite of data types	1	2	2	1	2	3	1	3	3	3
Agreed SLAs and protocols	2	3	2	1	2	2	1	3	2	2
Resilient to cyber attacks	3	2	1	2	1	1	3	2	1	2
Security classifications for data	1	2	2	1	2	2	1	2	2	2
Integrate variety of sources to produce new information	1	1	2	1	1	3	1	1	1	2
Knowledge generation through data discovery and exchange and analysis	1	2	3	1	2	2	1	2	2	3
Total Score	**21**	**23**	**34**	**19**	**23**	**34**	**18**	**25**	**25**	**35**

Key	
Barely supports	1
Supports to an extent	2
Supports well	3

Figure2 shows a summary scoring of different aspects of a software architectural style against the key requirements for the RCM data architecture identified in the Review of Current RCM Developments.

The aspects of architectural style used in this analysis are:

Interaction Style: what is the overall structure of the architecture?

- Stand-alone: there is no direct interaction between participants. Systems are independent of each other; data exchanged by file transfer only
- Client Server: there is a shared database with client systems connecting to it. Clients may have more or less processing capability
- Bus-based: systems are connected by a data interchange bus. Once connected to the bus, a system can interact with any of the other systems also connected

Data Management Style: where are the data being stored?

- Silo: each system has its own independent data store, not shared with any other
- Central database: there is a single central data store where all data are kept
- Federated: there is a set of data stores, but a linking mechanism allowing data to be queried from any one without knowing which data are stored where. A broker mechanism is necessary to route queries to the right place

Processing Style: where is the data processing done?

- Silo: each application does its own processing
- Application Layer – Centralised: client-server arrangement, where the 'business layer' is tightly associated with the server, not on the client. Also known as 'thin client'
- Application Layer – Distributed: client-server arrangement, where the business layer is associated with the client rather than the server. Also known as 'fat client'
- Service-based: processing elements are separate from applications and their users and are connected to via a network. They may be offered as web services using a SOAP protocol or similar

In this figure, the higher total score suggests a better overall fit with the requirements.

On this basis, an architecture with **bus-based interactions**, using a **federated arrangement of databases** and with processing done using a **service-based**

approach would appear to match the requirements the best. This describes a message-oriented bus-based architecture of loosely-coupled systems. The discussion which follows assumes this basic structure.

Principles of the data architecture

Introduction

From the analysis in the preceding section we can define the key principles which guide the proposed RCM data architecture. It is a service-based one, based around a shared set of data models, data services and message-based interchanges. In this section, we describe the components of the data architecture and outline how it can be used in data sharing projects of different levels of sophistication; later sections go into more detail.

Components of the architecture

Previous good practice and the key ISO standard 19526 identify a set of core components necessary to define a robust and usable data architecture that enables this service-based approach to be taken. These are:

- A set of shared data models which define a common view of the information to be exchanged, covering basic data items, RCM data and rail data, at different levels of abstraction from conceptual down to specific data formats (Shared data models section)
- A shared reference data repository to hold common reference data items (Shared reference data section).
- Shared IT components to support the architecture and provide some of its functions (Shared IT elements section).
- A means of sharing metadata (data about data): ownership, provenance, quality, description (Metadata: ownership, responsibility, conditions, quality section)
- An agreed method of querying for data and providing responses (Referencing, requesting and querying section)
- A security model (Security section) to safeguard users' data and systems and to guarantee data integrity.
- An extension mechanism to enable users to add their own data items to architecture-compliant interchanges (Extensibility section)

Developing and using the architecture

It is neither possible nor desirable for the data architecture to be built in its entirety before being made available for use. It will need to evolve and develop with use. Nor will every RCM data integration project require all the features that the architecture will ultimately provide, some of which will require significant central investment.

We have suggested a set of 'integration levels' (See Integration levels section) which correspond to types of data integration RCM projects may wish to employ. For each level we consider the amount of IT infrastructure and shared data it requires: how much time and effort will be necessary to provide it.

Shared data models

Data integration relies on all participants in data interchange having a consistent way to represent the aspects of the world they are dealing in. One of the key elements of a data architecture which supports this goal is a shared understanding of the data involved. This is the purpose of the data models. The models exist independently of any actual implementation or data interchange but can be used to standardise all such implementations.

Several aspects of these models must be seen for now as provisional, since they depend on ways of representing industry data that are defined elsewhere and in the middle of a phase of evolution and innovation. In particular, the way that linear assets are referred to will be defined by the ORBIS programme; and rolling stock by a combination of LINX-TM and the replacement for the Rolling Stock Library.

Hierarchy of data models

There is no single data model for everything associated with rail data interchange. Instead, there is a hierarchy of data models which range from highly-specific ones associated with individual industry processes and their IT systems, right up to the most abstract and general conceptual view of 'assets' and 'measurements'.

In order to be generally applicable, the data architecture must have an abstract conceptual data model that can be used for any type of asset, any type of measurement or data and any type of RCM-related application.

However, in order to be useful to actual data interchangers, it must also be rooted in the types of asset, standard representations and data formats that they understand and use regularly.

Figure 3 - Hierarchy of data models

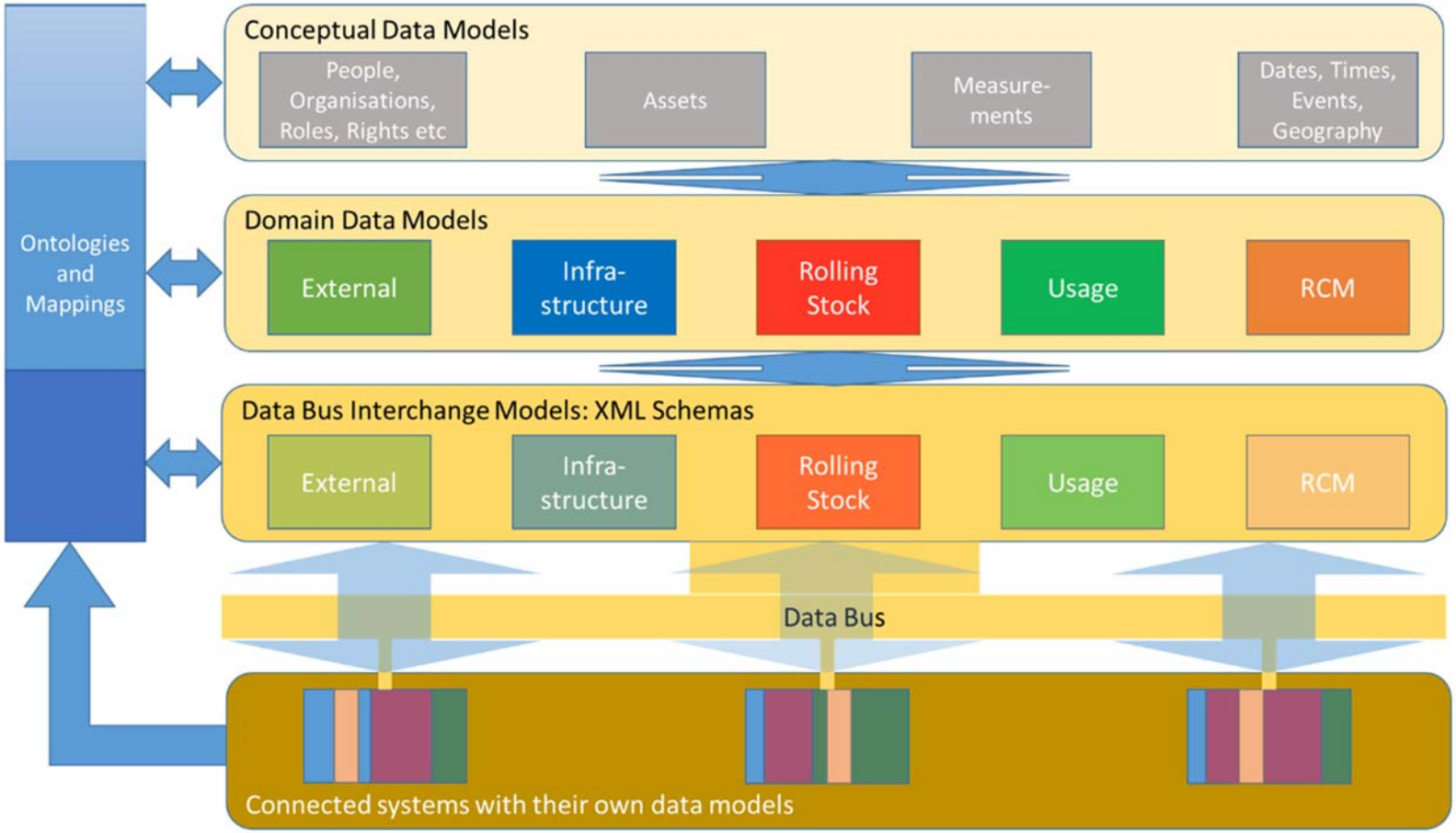


Figure 3 shows the hierarchy of data models in the RCM data architecture. It also shows an important possible role of a Rail Ontology in managing the mappings between the layers.

The 4 layers of data model in this figure are, from the top down:

- Conceptual data models. These deal in the abstract entities associated with asset remote condition monitoring – assets and measurements about them, as well as more general concepts required to manage data about places, people, rights and roles. At this level there is nothing specific about the rail industry in the data model. Key sources for these models are ISO standards and global standard data models and ontologies such as the Dublin Core
- Domain data models. These represent the railway and other signification application areas such as RCM in a way which is independent of any specific IT system, business process or data interchange requirement. They are informed by existing shared standards such as RailML and RailTopoModel, TSIs and Railway Group Standards
- Data bus interchange models. These models reflect data structures appropriate for data interchange. They are geared to clear and concise expression in terms of XML Schemas for message types, which can themselves be expressed physically in different standard formats such as .csv or JSON. The key driver for these is that they are easy to use: they contain familiar concepts in a clear structure, supporting the types of data interchange required by users. These data models form the 'lingua franca' for a data-bus model of data interchange through data services
- Connected systems' data models. These are the data models used by actual IT systems which will provide or consume RCM data. They therefore sit outside the scope of the data architecture, but their design should inform that of the Data Bus interchange and higher models. Each such data model will be an amalgam of elements of infrastructure, rolling stock, usage and RCM data, using its own concepts and data formats

Mapping between data models

There is a conceptual linkage between the layers. Each lower layer is a 'realisation' of the layer above; each higher layer is an 'abstraction' of the layer below. The entire structure is built by a combination of top-down processes, in which a starting abstract model is applied to a particular domain; and bottom-up processes, in which real data models are compared and shared overlapping concepts identified and resolved.

The choice of abstract models at the top layer is based on previous work in this domain and others, where shared concepts and relationships have been identified, applied and agreed.

The shape of the models in the lower layers is based on understanding of the structure of the rail industry, its business processes and the need for data interchange to support them.

Each system which will provide or use data from the data architecture will need its sponsor to carry out the mapping between its own data model and that of the data bus interchange model. The code elements which do this mapping are known as data adapters.

In the example data models and datagrams presented in this document, which represent aspects of the top three layers described above, this mapping has been done 'manually' by the project team based on their understanding of the rail domain and of the data that is transferred. Any data interchange project will require the data elements it needs to be represented at the data bus interchange level: provided by the data architecture. The bottom-up mapping process and reconciliation with higher-level models will therefore need to be repeated each time a new type of data or interchange is encountered. There is thus likely to be considerable benefit to systematising and automating the process as far as possible.

The Role of ontologies

The process of mapping between data model layers is one of alignment of concepts based on an understanding of their meanings at the different levels of abstraction. Ontologies can represent these conceptual relationships in an unambiguous way, in a manner that a straightforward entity-relationship or UML style of data model cannot do, with the added benefit that the representations can be queried programmatically and so support the type of automation of the model mapping processes described in Section 5.2.2. In particular, the ontological approach can validate data models and detect omissions and inconsistencies in them, thus improving data model quality.

There is thus a good case for the development of a set of ontologies in parallel with the data models, and of an ontological approach to the concept alignment problem.

In Figure 3 we have shown a stack of ontologies alongside the data models, being informed by them and informing them. The layering of the data models agrees with the layering also seen in emerging rail ontologies.

Once this set of mapping ontologies has started to be built, it can also start to be used for other purposes, such as the derivation of relationships through inference based on the rules contained. This is a key enabler for the asset relationships level of data integration described in Section 5.2.

Details and requirements

Details of the data models proposed and the architecture requirements relating to them are in Section 6.

Shared reference data

Data interchange requires that the same names be given to the same things – that there is a shared vocabulary. It will build upon the shared data structures described above. This vocabulary will need to include the following elements:

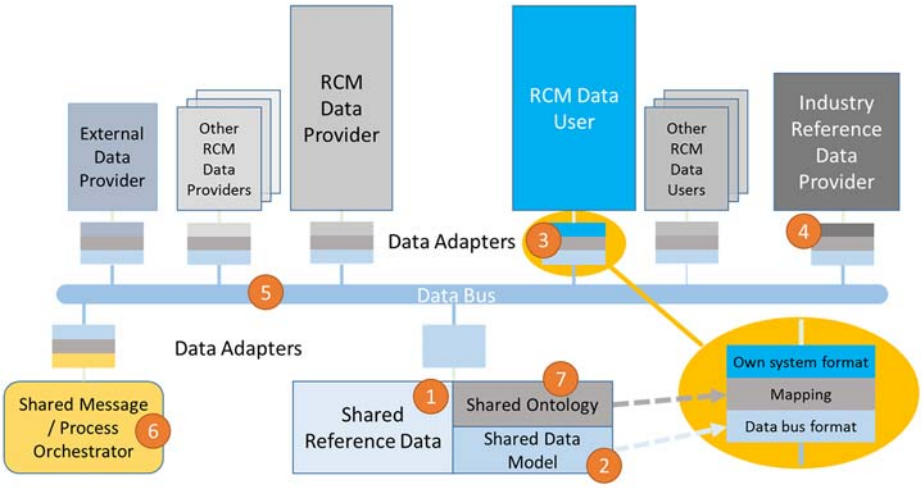
- Basic data format specification. This will contain the definitions of the data architecture's standard data interchange formats, in the form of references to existing standards, XML schemas and guidance documentation.
- RCM data format specification. This will contain the definitions of the RCM data types at the levels of the ISO 13374 stack and will therefore formalise data interchanges in these terms.
- Reference data. This refers to all the types of lookup data about the configuration of the data architecture, RCM vocabulary and the rail network that are shared with users to facilitate data interchange. It also includes a catalogue of web services available to users to query Web Services made available by industry reference data systems. Also considered are 'reference data proxies' which hold copies of data from industry reference systems to reduce the demand on the reference systems themselves.
- Web service definitions. This refers to a more general set of data specifications required to support general data interchange via web services. These definitions collectively define the data bus, since they inform the development of data adapters.
- Messaging configuration. This stores information about the configuration of messaging services to support cross-industry business processes involving multi-stage data transfers.

Details of the reference data elements and the architecture requirements associated with them are in Section 8.

Shared IT elements

Each type of data integration requires some support from shared IT components. The main IT elements of a shared service-oriented bus-based architecture are shown in Figure 4.

Figure 4 - IT components of data architecture



These are marked with orange numbers. The sequence of numbers roughly corresponds to the order in which the elements need to be provided to support data integration at different levels:

1 Shared reference data store. At its simplest, this is a web-hosted repository of data standards and agreed formats; it then develops into a database of shared standard lookups, then configuration data.

2 Shared data model store. This is a web-hosted repository of XML schemas and the domain data models that they represent, plus documentation.

3 Data Adapters. At the Reference Data and Datagram Formats levels of integration, each IT system which wishes to interchange data using the architecture must map its own internal data formats and structures into the standard ones defined. Data is converted to the agreed format by the data provider, and from that format by the data user. This is the essential element of a Data Bus.

4 Reference Data Providers. The sponsors of the master data repositories in the rail industry for asset and train service data must make reference data available using Web Services in the Data Bus format. This enables all data

sharers to have up-to-date and accurate data on assets, their names, structure, position, status etc.

5 A Messaging Bus. This is a framework which enables data providers and users to communicate by messaging. The messaging bus adds features such as guaranteed delivery, broadcasting and publish / subscribe services.

6 A Message Orchestrator. This enables complex cross-industry business processes to be defined in terms of interactions between connected systems offering and consuming web services.

7 An Ontology data store and compatible data adapters. This makes available the shared model of the rail network defined using ontological rules, which supports the development of more sophisticated data adapters and new software tools such as reasoners.

Metadata: ownership, responsibility, conditions, quality

Data suppliers may wish to identify themselves as owners or creators of the data that they share or may need to apply licensing or usage conditions to it; data consumers may wish to know its provenance and details of its applicability to their own usage; Railway Group Standards or other legal stipulations may require that the parties responsible for data or actions on it are identified.

All of these concerns are handled in the RCM data architecture by the use of metadata – additional data items which describe aspects of the asset and RCM data itself. The XML schemas in which the data interchanges are specified can have any of the following metadata items attached to each element:

- Ownership, rights, responsibility, restrictions, and licencing information using the standard PROV method [8], optionally including standard metadata terms defined in the Dublin Core [9], mapped using the official mapping [10].
- Data quality information, provided using either MIMOSA-based absolute definitions of tolerance, precision, signal-to-noise, or application-specific indicators of dependability.

Referencing, requesting and querying

The RCM data architecture will enable many different types of IT system to interchange data, each with its own internal data structures and data access mechanisms. To support a standard mechanism for enquiring of asset data and the associated RCM data, the architecture provides two important mechanisms:

- A standard for referencing segments, assets, sensors and the RCM data associated with them. This is a URI-based scheme, where each object would have its own 'web address', as would key data associated with it
- A standard for requesting data about these objects, with common selection parameters such as date / time range, data type, location or proximity
- Standard ways of representing the data interchanged. These are defined in the form of XML Schemas, but actual data being interchanged can be represented in a number of different ways provided they obey open standards and are compatible with the schemas

The exact form of the referencing schema will only be able to be defined fully once the designs of the master data systems have been clarified. These will be ORBIS for fixed and linear assets and the replacement for the Rolling Stock Library, for moving assets. The schema for sensors attached to assets will depend on the presence of a master ID mapper.

Security

Cyber security is growing in importance as the use of connected IT systems and COTS equipment grows in the rail industry, along with a growing level of threat. Since the data architecture enhances this connectivity and promotes the use of open standards, it must ensure that it does not have a negative impact on the overall security of UK rail information systems.

Industry guidance on cyber security is held on the 'Cyber security in technical systems' area of the RSSB website [11].

The RCM data architecture supports the security of data and of connected systems by defining these mandated behaviours:

- The use of Secure HTTP (shttp) as a secure protocol for data exchange
- The ability to sign data files and messages using the XML Signature [12] signing standard

- The ability to encrypt data files and XML messages, or elements within them, using the XML Encryption [13] standard
- Support for independent authorisation of systems and users accessing connected systems
- Access to connected systems via published message-based gateways / end points only
- Precautions against 'injection' attacks, where queries against systems or databases can include hostile executable instructions inside their query text

Extensibility

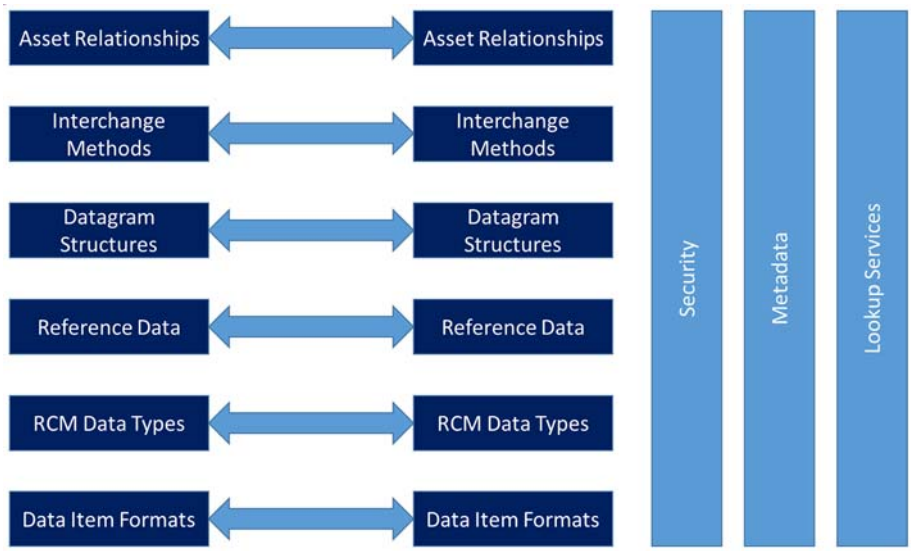
The types of asset, sensor, data, process, application, format used in the UK rail industry can all change over time. Specific data items relevant to particular applications may need to be transferred via the RCM data architecture even though they fall outside the defined standards.

The architecture provides extension points within its XML Schemas to allow additional data items to be added for specific purposes. The content or format of such extensions is not validated or controlled by the architecture, but the data transport mechanisms will ensure that they are transmitted with the same levels of security and reliability as the standard data.

Integration levels

The data architecture has a layered structure, where the layers correspond to increasing levels of data integration and thus increasing opportunities for business process improvement. The layers are shown diagrammatically in Figure 5 and described in Table 4. The lowest level of integration is at the bottom of the figure.

Figure 5 - Data architecture overview



The lowest layers require little supporting infrastructure or central development work and so can be implemented easily and quickly; the higher the layer, the more central organising work is necessary.

Table 4 - Proposed data architecture layers

Data architecture layer	Standardisation level	Integration capabilities; dependencies; limitations
1 Standard data item formats	Basic: common understanding of standard quantities.	Simpler coding and more reliable interpretation of data from other parties. No standardisation of asset or RCM data, or of file formats or interchange mechanisms.
2 RCM data types	Structured: RCM data at all levels of the ISO 13374 stack	Standard representations for all types of alert or alarm, sensor, health, prognostic data. Enables RCM services to be set up with clear process/data interfaces.

Table 4 - Proposed data architecture layers

Data architecture layer	Standardisation level	Integration capabilities; dependencies; limitations
3 Standard reference asset nomenclature	Asset-level: data from different sources about the same asset can be merged easily.	Significant step forward enabling a number of new processes and new data integrations. Will need support in terms of referencing services and asset ID repositories and lookup functions (such as AVI or location referencing). No standardisation of file formats or interchange mechanisms.
4 Datagram structures	Shared understanding of formats for data interchange: files, messages, other.	Standard datagram formats for all kinds of RCM data. Significant level of integration possible at this level, at least for asset health and history; and the ability to drill into RCM data from multiple sources in a standard way.
5 Interchange methods and transaction protocols	Shared mechanisms for requesting and transmitting data.	The ability to request and receive data in real time and by a number of asynchronous methods. This enables integrated business processes involving RCM data to be set up. The more real-time aspects require a message-based infrastructure such as an ESB to mediate the interactions.
6 Asset relationships	Standard methods for expressing and enquiring of complex asset structures.	Enables roll-up of single assets into groupings (such as point motors -> interlockings -> route sections; or vehicles -> multiple units -> formations). Enables drill-down into assets' components (such as vehicle -> bogie -> wheelset -> bearing). This enables higher level RCM functions such as the assessment of network capability in the presence of faults or restrictions of some assets.

Each of the integration levels described in Section 5.2 requires some supporting data and IT infrastructure, starting from simple requirements at the 'data item formats' integration level, going up to a considerably higher requirement at the asset relationships level.

Figure 6 - Architecture data and IT components

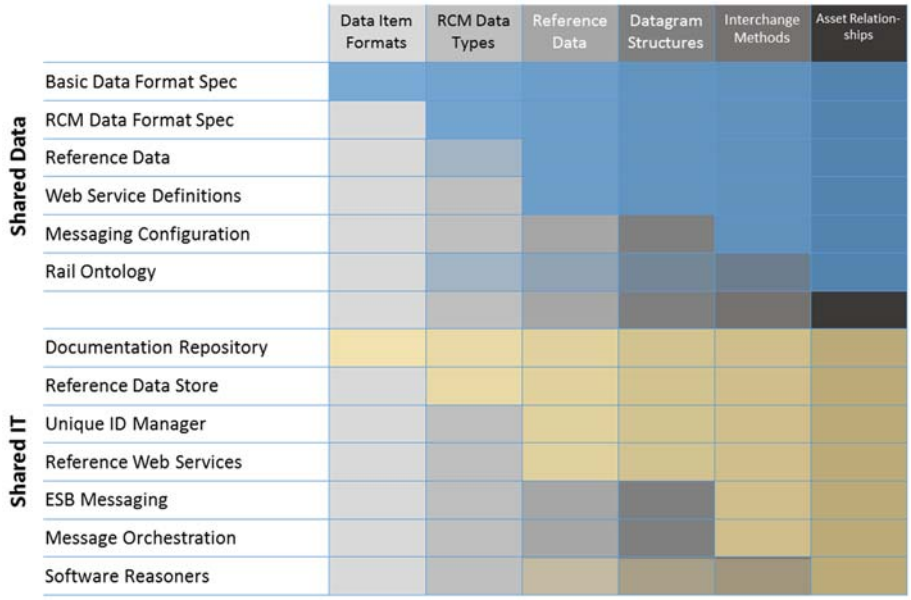


Figure 6 shows the main data and IT elements required for each of the six integration levels. The heavily-shaded blue and yellow squares indicate where the element is necessary to deliver an integration level; lightly-shared squares indicate where the element would add functionality but is not a pre-requisite.

The main message from this figure is that data integration efforts can be supported at the lower levels of integration with relatively little central investment in IT and relatively little top-down standardisation or data modelling work. Features can be added progressively as the demand for integrated data and standardisation increases, based on successful implementation of the simpler elements. Integration at the higher levels is built upon work previously done to support lower levels.

Data models

Introduction

In this section we consider the data associated with the data architecture.

The data models presented in the sections below cover the following categories of data associated with RCM on rail assets:

- At the Conceptual level (Section 6.3.2):
  - Fundamental concepts: segments, assets, agents, measuring locations, data types
- At the Domain level:
  - Railway network topologies: the way sections of route are connected and related to each other; the relationship between the physical structure of the network and the other views that can be taken of it for operational or commercial purposes (Section 6.4.2).
  - 'Point' fixed railway assets: single items at a single location, such as switches, signal cabinets, structures (Section 6.4.3.1).
  - 'Linear' fixed railway assets: items extended over a distance, such as track, OLE, foundation (Section 6.4.3.2).
  - Moving railway assets: rolling stock and their components; operational trains (Section 6.4.4)

Section 6.6 describes the role Ontologies can play in the management of data models and actual data.

The Data Interchange formats and XML Schemas are discussed in Section 10.2.

Conceptual data models

These models reflect a highly-abstracted view of the world which is independent of any particular type of application. They provide a unifying backbone on to which the more specific domain-level models can be built.

The models are based on ISO 15926 and in particular, its interpretations by MIMOSA known as OSA-CBM [14] (a data interchange specification) and OSA-

EAI [15] (a shared data model to support information exchange). These provide a suitable framework for defining the fixed and moving parts of the rail network, the physical objects providing these parts at a given time, the relationships between them, the RCM equipment which monitors them, the data that result and the processes which are carried out on the data.

MIMOSA Registry data model

The core abstract concepts required by the data architecture are included in the Registry data model shown in Figure 7. This is one of the 28 sub-models which make up the MIMOSA CCOM (Common Conceptual Object Model) data model. They are described in the following sections.

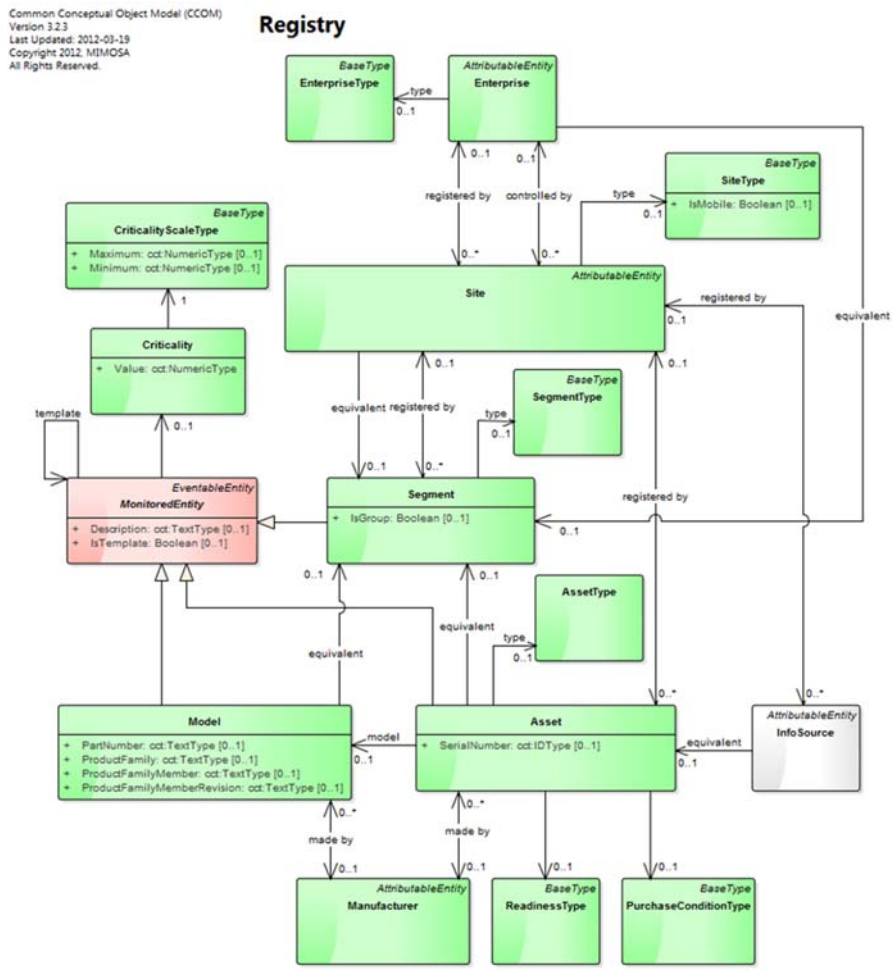
Organisations and databases

Data within the scope of the RCM data architecture can be sourced from and used by a large number of organisations, elements of hardware and people. It can be stored and manipulated by different IT systems with their own data stores. Organisations can have complex structures, with data passing between different sectors or offices within the same organisation as well as across organisational boundaries.

To keep track of all of this, MIMOSA OSA-EAI defines a registry with 2 types of entity in it – enterprises and sites. The main purpose of the hierarchy is to ensure that a site has a unique identifier and that some authority (the enterprise) has the ability to create new sites and assign unique identifiers to them.

A site in MIMOSA terms is a structural unit which is responsible for a grouping of assets and can organise the issuing of unique identifiers (in the form of UUIDs – see section on unique identifiers) for them. In track terms, it may correspond to a permanent way engineer's area of authority, a single interlocking or the domain of a single computer system; in rolling stock terms to a rail vehicle. It is uniquely identified either by a UUID issued by MIMOSA, or by the combination of a User ID issued by MIMOSA and a User-defined ID.

Figure 7 - MIMOSA OSA-EAI CCOM registry data model



An enterprise would be an owning body such as Network Rail, a ROSCO, or an IT system supplier.

In the intelligent infrastructure implementation of MIMOSA OSA-CBM, each data logger is equivalent to a site and each equipment supplier an enterprise.

From the RCM data architecture point of view, it is important only that every data item has a unique site identifier.

0001 Users of the RCM data architecture must ensure that every data item is identified by a globally-unique site identifier.

0002 The RCM data architecture should support a registry of site identifiers to simplify their association with specific data sources.

Unique identifiers and registration

The railway network is comprised of entities which persist over time but may undergo many changes through their lifecycle. Typical changes may be to name, exact position, relationship with other entities, owner.

The same railway entity may be referred to in different ways in different contexts, such as operational or engineering spheres; or different computer systems. For example, there are at least 4 different ways to represent locations on the railway; and at least 3 to represent a train.

Where data needs to be gathered about an entity over its life history, or shared between different parties who refer to it in different ways, it is essential to be able to identify that entity unambiguously. This is done by means of a unique identifier.

Allied to the use of unique identifiers is a translation mechanism between an entity's unique ID and the various system-specific or user-specific identifications of the entity: this service allows the entity to be looked-up and referred to by any system connected to the architecture.

MIMOSA CCOM has the concept of a Unique Identifier which is defined to be a UUID in the sense of RFC 4122 [16] (see Section 7.3 for details). It is associated with any persisting entity when it is created or made known to the registry, and does not change through that entity's life even though any other descriptive identifier of the entity may change.

MIMOSA's Structured Digital Asset Interoperability Register (SDAIR) [17] is one implementation of the translation mechanism. Its data structure allows a given UUID to be associated with representations in any number of other systems, and each such representation to have additional properties which can be used to help identify it.

0003 The data architecture must support a UUID for any entity.

0004 The data architecture should support a registry function similar to the MIMOSA SDAIR to issue UUIDs and manage descriptive identifiers for them. This may consolidate or link existing industry UUID registries, as well as storing system-specific references for any entity.

People and processors

At the state detection level and above of the ISO 13374 stack, the processing which generates data events is not done by sensors and loggers, but by organisations, people and software components. MIMOSA abstracts all of these notions as agents.

Organisational and ownership structures and the jobs that agents do are represented in MIMOSA by roles and agent-agent relationships.

As far as the RCM data architecture is concerned, the important aspects of an Agent are that they are:

- identified by a UUID
- associated with any data event above the level of state detection

005 The data architecture must enable every Data Event to be given a UUID representing the sensor, logger, person, organisation or software component responsible for it.

006 The data architecture should support the use of a reference or lookup function to find a UUID for a given sensor, logger, person, organisation or software component.

007 The data architecture should support the management of RCM data reference and lookup data.

Segments and assets

In considering the maintenance and condition of rail assets, it is important to distinguish between the designed object (such as outer left-hand wheel bearing of the front axle of the front bogie of a rail vehicle) and the specific object fulfilling that role at a point in time (SKF bearing, type XYZ/1, serial number 3022315, fitted 23/01/2012). The designed object persists for the life of the whole ensemble (vehicle in this case); the specific object may be changed over time, may be maintained or overhauled, and may be moved to fulfil the role on a different designed object.

MIMOSA OSA-CBM uses the term Segment for the designed object, and Asset for the specific item fulfilling its role at a point in time. In the words of MIMOSA:

> *'An object is an Asset if it meets one of these criteria:*
>
> - *Could be depreciated in a financial system*
> - *Could be tracked by serial number*
> - *Could be transferred/sold and utilized/installed at a different Segment possibly associated with another Site at another Enterprise'.*

In most current RCM systems, RCM data are gathered for a segment – for an identified location such as point motor on switch SN210, or outer left-hand bearing on the front axle of the front bogie on vehicle 27154. The identification of the specific asset – serial numbered item – is not known at the time the measurement is gathered. However, making this connection is an important enabling factor for the higher-level RCM functions such as prognostic assessment and advisory generation: it is impossible to track the degradation of an asset over time in an RCM system if the system is unaware that the asset has been replaced during the time period.

The relationship between assets and segments is normally held in asset management or component tracking systems rather than RCM systems themselves. The RCM data architecture can facilitate the interchange of this type of information by supporting standard nomenclature and lookups between segments and the assets fulfilling them at a particular time.

0009 The data architecture should support the lookup of specific assets associated with a segment.

Segment and asset relationships

Railway assets, whether infrastructure-related or rolling-stock-related, exist in complex structures with complex relationships of composition (asset A is made up of sub-assets B, C) or association (asset D overlaps with assets E, F; asset G can only operate on assets of type H). An important aspect of any IT system which uses RCM data is to be able to follow these relationships to identify affected components or to draw route-wide or network-wide inferences from asset-level evidence.

The reference data about the railway network and its components, and about rail vehicles, their components and their groupings, are held in different master reference systems. There is significant change occurring: new master systems are being proposed for both fixed and moving assets; and new methods of looking up the reference data they contain are being considered. Importantly, asset data provision is being seen in terms of services.

The architecture will support the types of relationship that occur between railway assets. Presently, there is no clear standard method for representing these relationships: it is hoped that the cross-industry RCM initiative may help to stimulate effort to provide one.

To this end, the RCM data architecture defines a standard way for important categories of asset hierarchy to be represented; and for RCM data at any of the ISO 13374 levels to be attached to them. This standard is based on the

ISO 15926 representation, as implemented in the MIMOSA OSA-CBM model [14]. (See subsections on Fixed railway assets and the network, and Moving assets.)

The architecture also defines a standard set of queries related to assets and their relationships with each other, implemented as request and response message pairs, which might form the basis of an industry-standard methodology for this type of enquiry (Section 12).

It should be noted that this area is one where the emerging technology of web ontologies could deliver value. Although not strictly part of the RCM data architecture, it could form the underpinning of the data structures which enable the asset relationship queries mentioned above to be satisfied. This and other uses of web ontologies are described in the section Industry reference data requests.

Domain data model - RCM data

RCM data and processes: ISO 13374

The data architecture reflects the 6-layer hierarchy of RCM-related processes set out in ISO 13374. This means that data transferred using the data architecture will abide by the following principles:

- All data elements can be characterised as being the outputs of one of the ISO 13374 processing layers or blocks. Each layer or block has a characteristic output:
  - A data acquisition (DA) block generates a scaled digital version of sensor data from a specified sensor, in one of a number of standard formats, with timestamps and quality indicators
  - A data manipulation (DM) block generates descriptors of the sensor data, for example: extracted features, conversions, integrations, filtered versions, normalisations, averages, statistical summaries, with timestamps and quality indicators
  - A state detection (SD) block generates indicators of the asset or sensor state, indications of proximity to, or exceedance of, threshold boundaries, identification of data or sensor anomalies, with timestamps and quality indicators
  - A health assessment (HA) block generates a health grade or a diagnosed failure mode for an asset, optionally with probability assessment and justification

- A prognostic assessment (PA) block generates a projected future health grade or time to failure in a given mode for an asset, associated with a set of future load or exposure values and with an optional explanation
- An advisory generation (AG) block generates a work request or operational advisory for action by external operations or maintenance systems

- A block in each layer can request data from blocks at the same or lower levels
- Each block has a defined protocol – an Application Program Interface (API) - by which data can be requested from it, and a defined set of formats in which the data can be provided

This approach means that both existing and new IT systems can be categorised according to where they fit in this layer and block model.

Figure 8 - Systems, APIs and the ISO 13374 layers

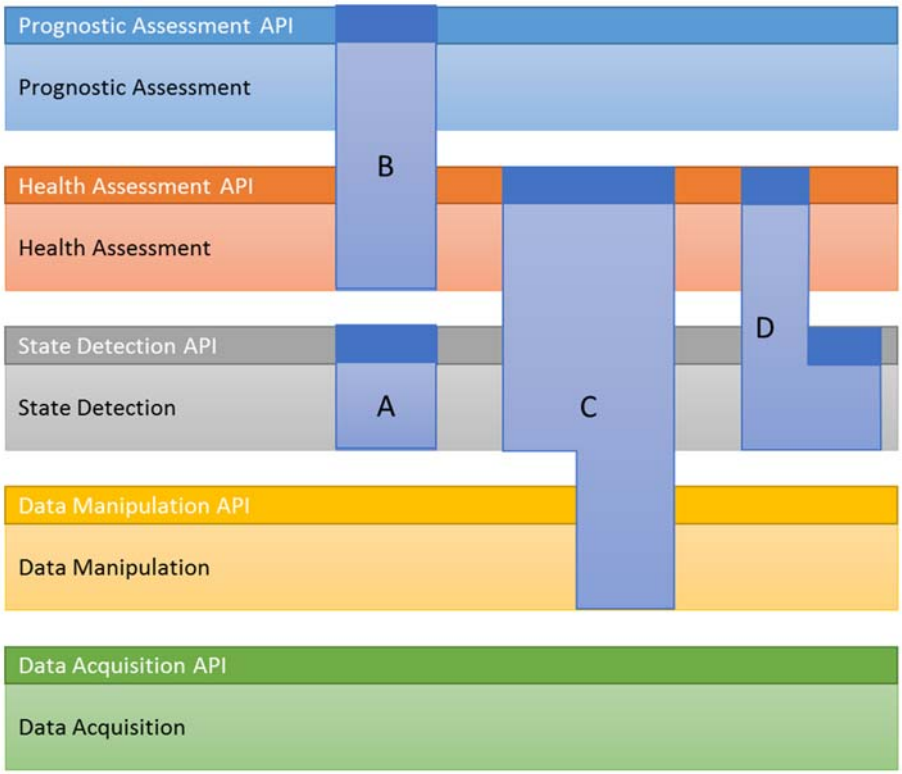


Figure 8 (taken from ISO 13374-2) shows how different IT systems in the RCM domain can carry out functions and offer data about different ISO 13374 processing layers:

- System A only does state detection functions and exposes just a state detection API. This may, for example, represent a specialist algorithm which carries out a sophisticated composite analysis of different sensors to come up with an overall asset state
- System B operates at both the health assessment and prognostic assessment levels but only outputs prognostic assessment data. This may, for example, be a forecasting module which calculates current asset health (though doesn't output it) and generates forecasts of time to live based on expected future loadings. System C has a similar structure, though operating over the DM, SD and HA levels and emitting health assessment data only
- System D carries out both state detection and health assessment duties and can output data from both these levels

The data architecture supports the interchange of RCM data of a number of different types. The types of data recognised by the architecture will be those identified in ISO 13374, described in the Review of relevant RCM developments report [6]; the terminology for recognising assets and sensors will be in accordance with the MIMOSA OSA-CBM specification [14].

This MIMOSA specification forms the basis for all the RCM data elements described in the architecture. It has been chosen because of its compliance with open ISO standards, its breadth of coverage of all conceivable types of RCM data and all types of RCM-related business process, and its use already in Network Rail in the Intelligent Infrastructure programme.

The most basic RCM data interchange will be a reflection of current practice – the transmission of alerts and alarms and status information about specific assets – but standardised. This corresponds to the State Detection and Health Assessment levels of ISO 13374. The format of alerts and alarms will conform to MIMOSA OSA-CBM.

The data architecture additionally supports the exchange of more detailed sensor-based data – whether processed, at the Data Manipulation level, or raw, at the Data Acquisition level, of ISO 13374. All types of sensor data can represented using standard formats, based around the MIMOSA OSA-CBM standards.

The data architecture support the exchange of higher-level RCM data corresponding to the Health Assessment and Advisory Generation levels of ISO 13374. The actual content of data interchanges at these levels is going to be much more about asset management strategies and processes than RCM per se, and so outwith the scope of the RCM data architecture; however, the data architecture does mandate standard representations of the RCM-related data items.

Basic entities in the rail RCM domain

Measurements and data

RCM data is gathered by measurement of physical quantities associated with sensors, the processing of those quantities into electronic or digital form and the drawing of conclusions about status, health and impact from them.

The types of sensor, data produced, process, algorithm and conclusion are legion. MIMOSA OSA-CBM supports a standard method of recording and categorising these items so that information can be captured and transferred in a consistent way for a wide variety of RCM data types.

0009 The data architecture should use MIMOSA OSA-CBM as the basis for RCM data interchange.

Key MIMOSA concepts are ports, measurement locations, data events and the types of RCM activity described in ISO 13374.

Ports

A MIMOSA Port corresponds to a data output channel; and a Data Event (described below) to a single reading from that Port. Ports can be defined at any of the ISO 13374 levels, and so can correspond to data being generated as raw sensor data, processed data (such as smoothed, averaged, integrated, transformed to frequency domain), alerts or alarms, or the higher-level types such as Health Assessments or Prognostic Assessment.

The Port may be associated with computational logic which determines how it generates it output: this is accessible as a set of configuration data and parameters which may include thresholds and conditions. MIMOSA offers a standard mechanism for describing this configuration which may be used to provide supporting information for any data generated by the Port. It does not require the exact specifics of the algorithm to be revealed, but does require the name, type, version and owner to be populated.

Measurement locations

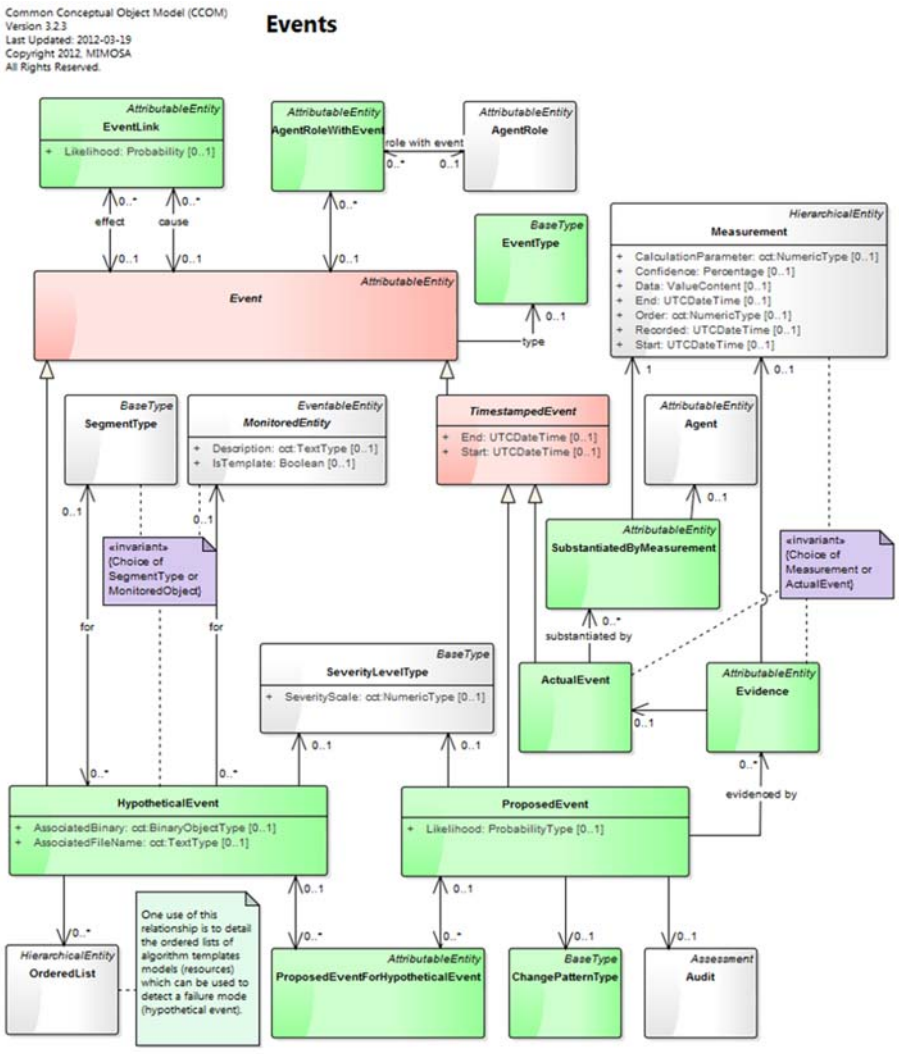
A MIMOSA Measurement Location corresponds to an output of a data collection device or sensor via a Port. It is associated with a Segment or an Asset. It has the following attributes:

- Unique ID
- Name or user-defined identifier
- Type, which defines what it is measuring. This might be a simple type such as Voltage, or a more complex one such as Proportional Amplitude of 4th Harmonic of Base Frequency.
- Unit of Measure
- Update frequency
- Details of the data source or transducer generating the measurement

Data Events

A Data Event is a single set of data generated by a Port. MIMOSA allows for a wide set of types of data to be passed via a Port and defines an XML schema for all of them. Figure 9 shows the MIMOSA OSA-EAI CCOM data model for events.

Figure 9 - MIMOSA OSA-EAI CCOM events data model



RCM data types

MIMOSA OSA-CBM defines data models and XML schemas for the data types described below, which represent the vast majority of RCM sensor-related data output types. It also allows, but does not recommend the use of, a mechanism for using user-defined data formats for any data interchange type which does not fit one of the other patterns.

Data acquisition

For the Data Acquisition layer of ISO 13374 (raw sensor data), the MIMOSA OSA-CBM data model is shown in Figure 10 and the data types and their typical uses in Table 5.

Figure 10 - MIMOSA OSA-CBM data model - data acquisition

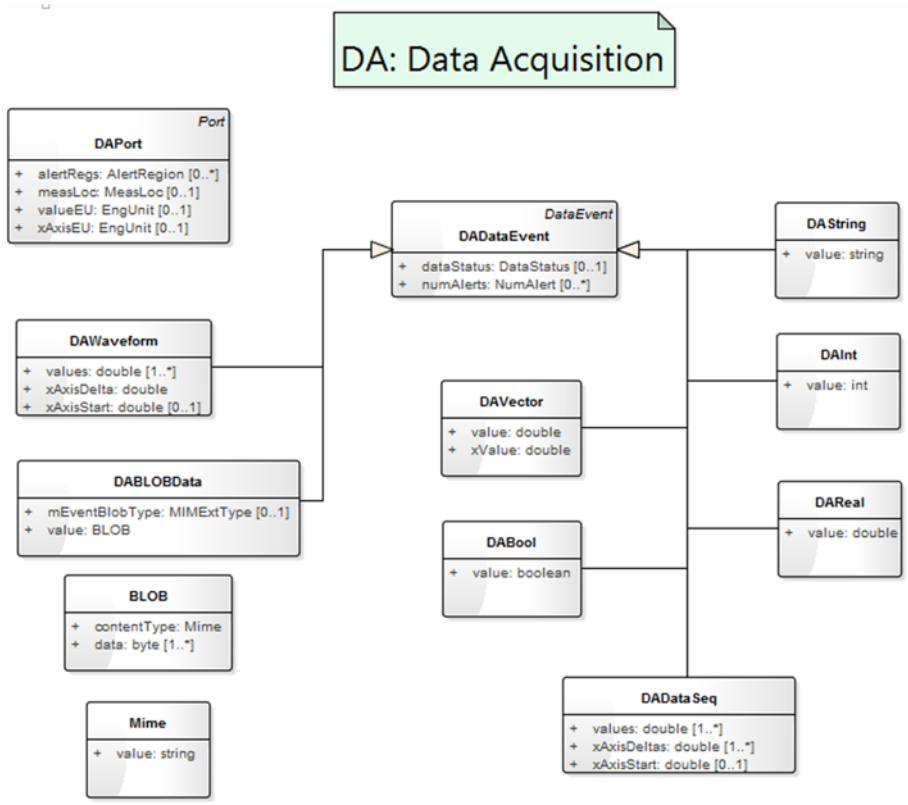


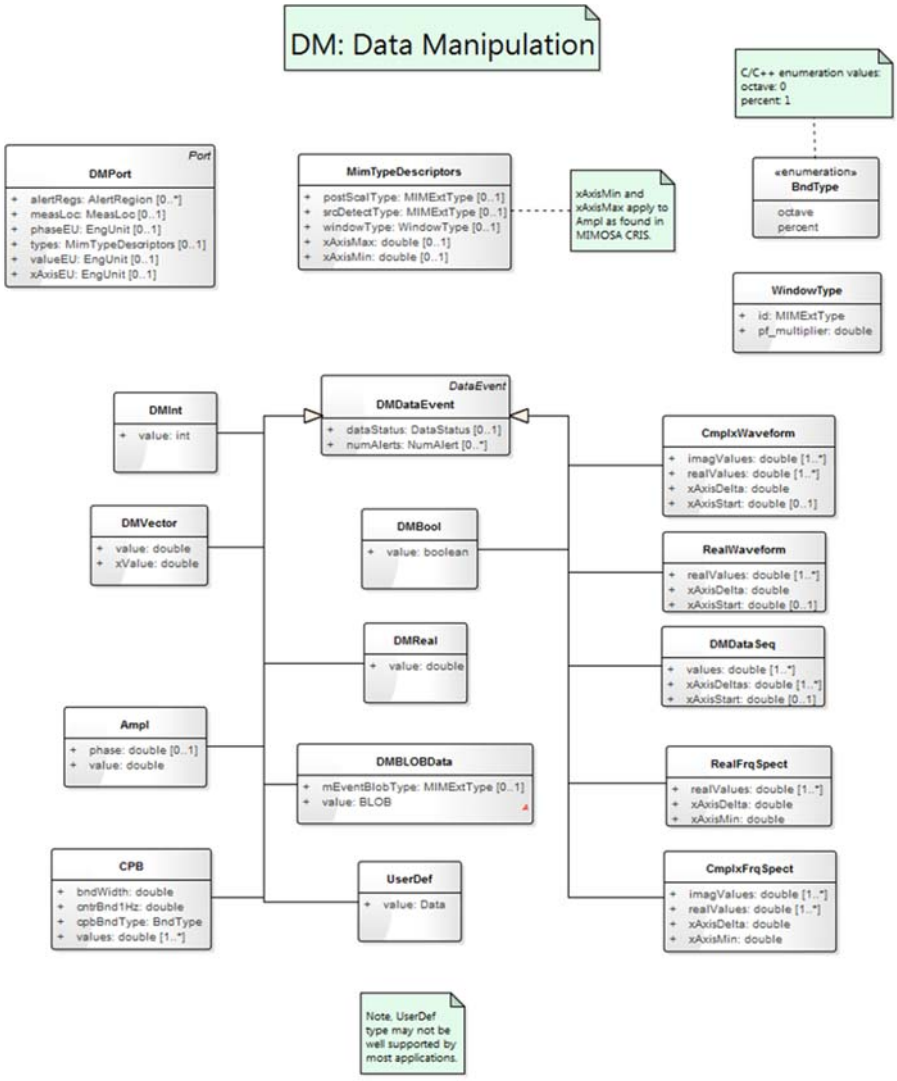
Table 5 - MIMOSA data types - DA layer

Data Type	Format	Typical Usage
Integer	A single integer	Count of occurrences e.g. revolutions
String	Alphanumeric string – no length restriction	Description such as 'Overheating'
Real	Double-precision floating-point number	Temperature in degrees Celsius
Boolean	True or False	Status = OK
Vector	A sequenced list of Real values	A set of temperature readings from a group of sensors
Waveform	A regularly-spaced list of Real values.	A set of current readings sampled at 100Hz for a point motor for a short period of time such as a point activation.
Data Sequence	A set of x, y pairs of Real values	A set of records of point operation, showing the start time and duration of each activation.
BLOB	A binary large object file identified by a MIME type	A photograph of an anomaly on an OLE neutral section; the sound file associated with a train passing over a microphone array.

Data manipulation

For the Data Manipulation layer of ISO 13374 (processed data), the data types are represented in the data model in Figure 11 and described in Table 6.

Figure 11 - MIMOSA OSA-CBM data model - data manipulation



Integer, String, Real, Boolean, Vector, Waveform, Data Sequence and Blob as for DA, plus:

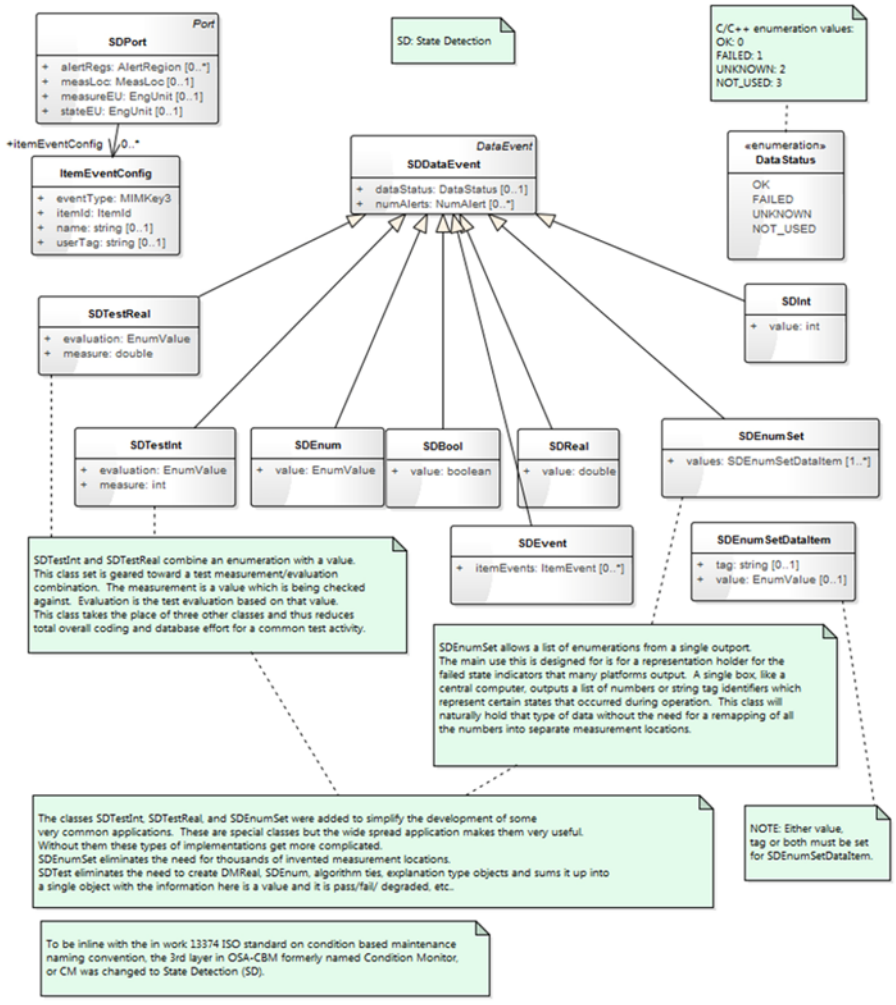
Table 6 - MIMOSA Data Types - DM Layer

Data type	Format	Typical usage
Ampl	Amplitude and Phase as Real values	Power Factor
Complex Waveform	List of Real and Imaginary components as floating point (Real) values	Electrical voltage and phase over time
Constant Percentage Bandwidth Spectrum	List of Real values and band start / width parameters	Noise spectrum
Real Frequency Spectrum	List of Real values by standard increment of frequency	Frequency spectrum such as that generated by rotating equipment
Complex Frequency Spectrum	List of Real and Imaginary values by standard increment of frequency	Phase response of acoustic equipment

State detection

For the State Detection layer of ISO 13374 (alerts / alarms), the data types are shown in the MIMOSA data model in Figure 12 and listed in Table 7.

Figure 12 - MIMOSA OSA-CBM data model - state detection



Integer, Real, Boolean as for DA, plus:

Table 7 - MIMOSA data types - SD layer

Data type	Format	Typical usage
Enumeration	Value from a list of valid values	Alert status: Normal, Alert, Alarm, Out of Service, Test
Enumerated set	List of enumerated values	As above, but for multiple statuses
Integer test	Integer value and enumeration	Measured value and the matching looked-up state: 16 clicks – OK
Real test	Real value and enumeration	Measured value and the matching looked-up state: 27.3V – Alert: Low

Health assessment

For the health assessment layer of ISO 13374), the data types are shown in the MIMOSA data model in Figure 13 and listed in Table 8.

Figure 13 - MIMOSA OSA-CBM data model - health assessment

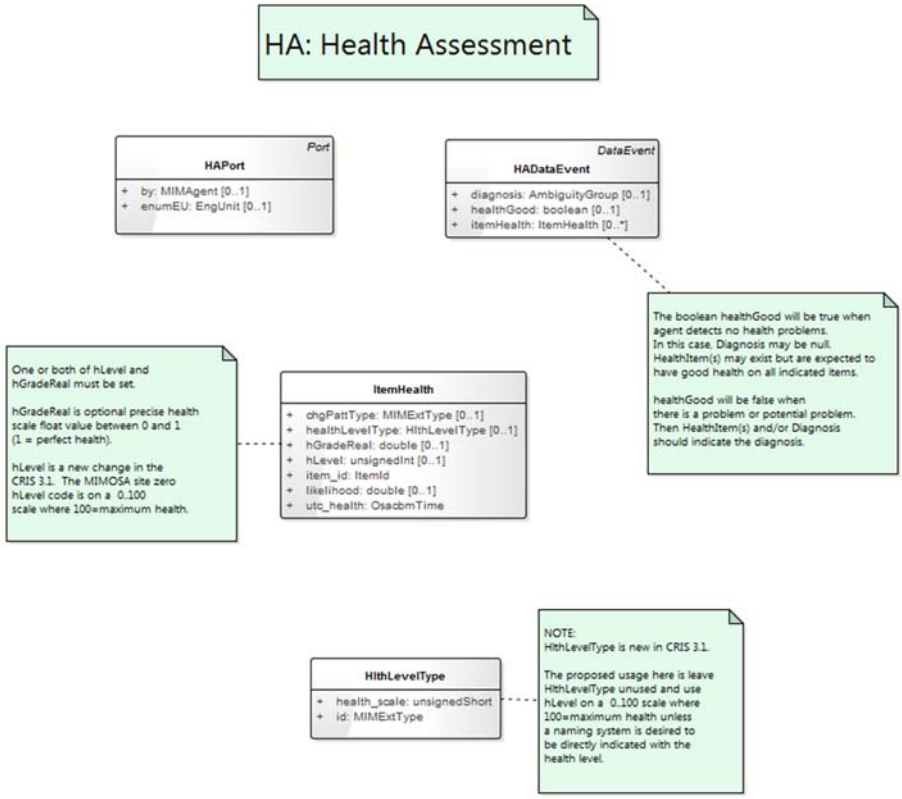


Table 8 - MIMOSA Data Types - HA Layer

Data type	Format	Typical usage
Health assessment	**Health level and type, expressed as Real (0-1) or Integer (0-100)**	0 = terrible health -> 100 = perfect health 0.0 = terrible health -> 1.0 = perfect health

Associated with the data items at this level can be additional diagnosis information which describes the reasoning for the health assessment.

Prognostic assessment

For the prognostic assessment layer of ISO 13374), the data types are shown in the MIMOSA data model in Figure 14 and listed in Table 9.

Figure 14 - MIMOSA OSA-CBM data model - prognostic assessment

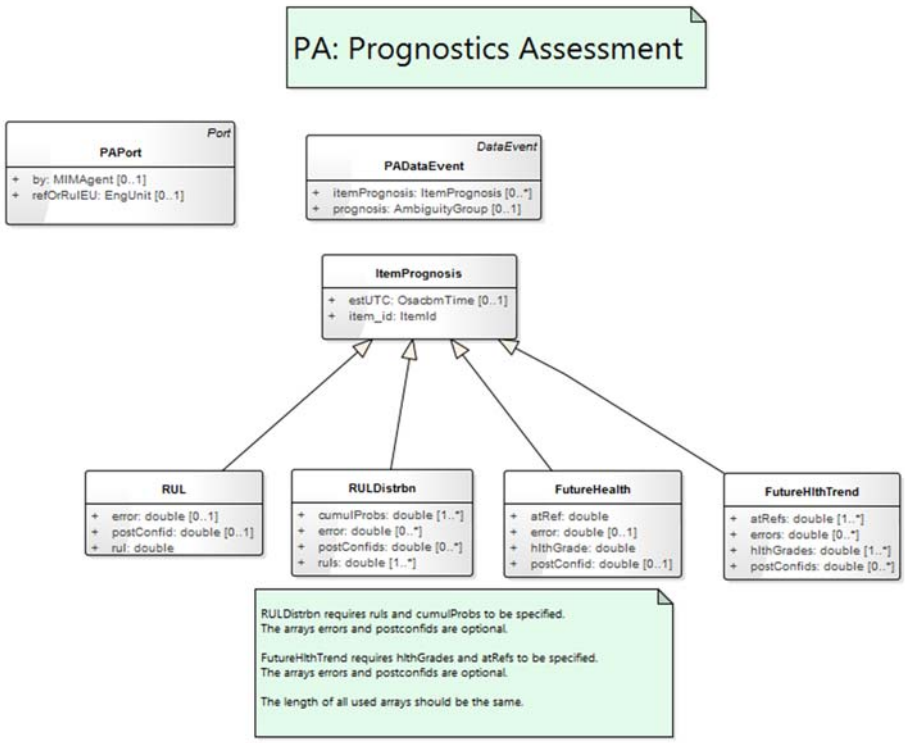


Table 9 - MIMOSA Data Types - PA Layer

Data Type	Format	Typical Usage
Remaining Useful Life	Estimate of life, confidence level	Time till asset will fail
Remaining Useful Life Distribution	Set of estimates of life and a cumulative probability distribution with error bars	Graph showing time till likelihood of failure exceeds a threshold
Future Health	Estimated health level at a specified future time, confidence level	Forecast health at a specific future time
Future Health Trend	Set of estimates of future health level for a sequence of future times, with error bars	Assessing when asset health will fall below a threshold

Associated with the Prognostic Assessment data can be additional diagnosis data which explains the assumptions behind the forecasts.

Advisory Generation

For the Advisory Generation layer of ISO 13374), the data types are shown in the MIMOSA data model in Figure 15 and listed in Table 10.

Figure 15 - MIMOSA OSA-CBM data model - Advisory generation

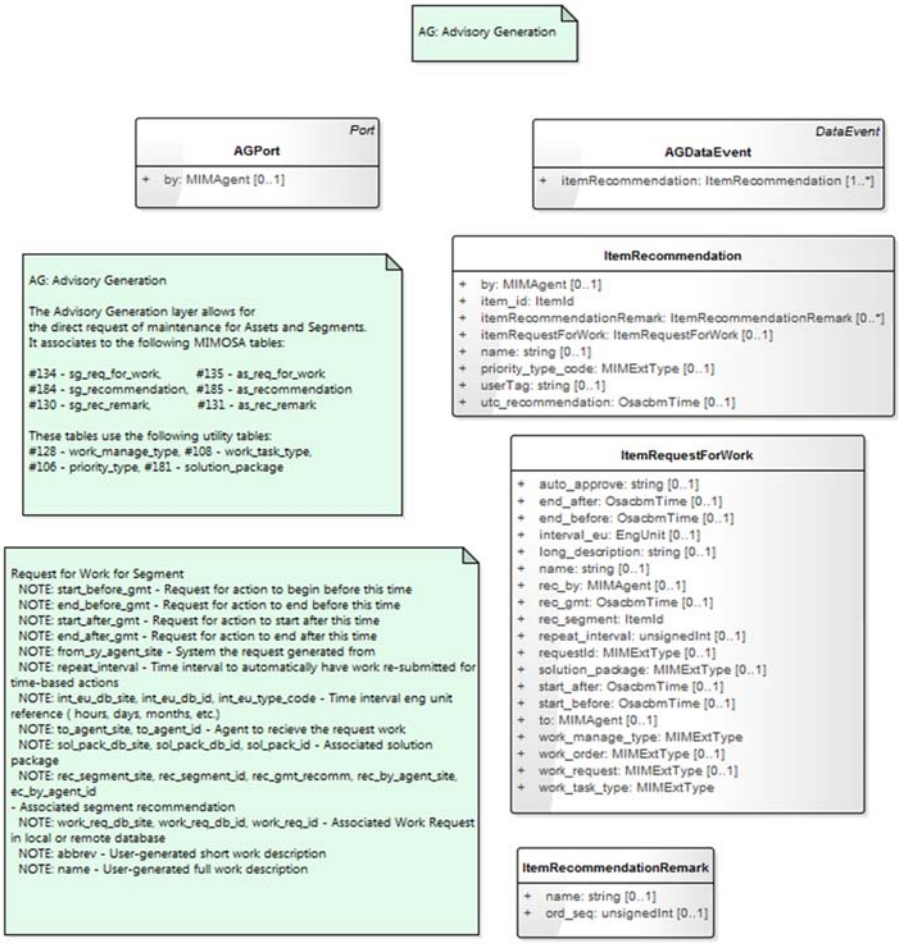


Table 10 - MIMOSA Data Types - AG Layer

Data Type	Format	Typical Usage
Recommendation	Priority, chosen from a list of levels, with explanatory remarks	Prioritisation of work on a large group of assets
Request for Work	Maintenance / Work Package Definition	Automatic request to work scheduling system to perform maintenance or correction on an asset.

In addition to these data types, MIMOSA allows (but does not encourage data) to be stored in a number of generic data types. These can be used for user-specific data types that are not supported by the standard MIMOSA types.

0010 The RCM data architecture must support all the MIMOSA OSA-CBM data types.

0011 The RCM data architecture must allow user-defined data types to be defined using the MIMOSA OSA-CBM generic types and compositing mechanism.

Domain data models – rail assets

Introduction

The data models in this section relate to the main categories of rail asset. Of course there are many different types of rail asset that may be subject to remote condition monitoring. It is not the task of the data architecture to define or restrict what these are, but it is to ensure that every category can be represented in the data architecture.

From a data modelling perspective, the categories of asset that need to be represented are:

- Networks and their topologies. Much of the analysis work related to rail RCM is about the impact of asset unavailability and failure on the capability and safety of the network. This type of analysis requires an understanding of the connected nature of the network and the relationships between the different views of it typically taken by engineers, timetablers and passengers. This is the domain of network topologies, covered in Section Networks and topologies.

- Fixed railway assets which are part of the infrastructure. These, covered in Section Fixed railway assets and the network, are subdivided into:
  - Point assets, which can be located by a single track location or geographical position. A switch, crossing or signal cabinet might be assets of this category
  - Linear assets, which extend along a route or track between two locations. A section of track or OLE would be of this category
  - Areal assets, which cover an extent of the railway that might comprise several route sections and locations. This category might include an interlocking or a large station
- Moving assets: rolling stock and the different views of operational trains they are used in. These are covered in Section Movingassest

Networks and topologies

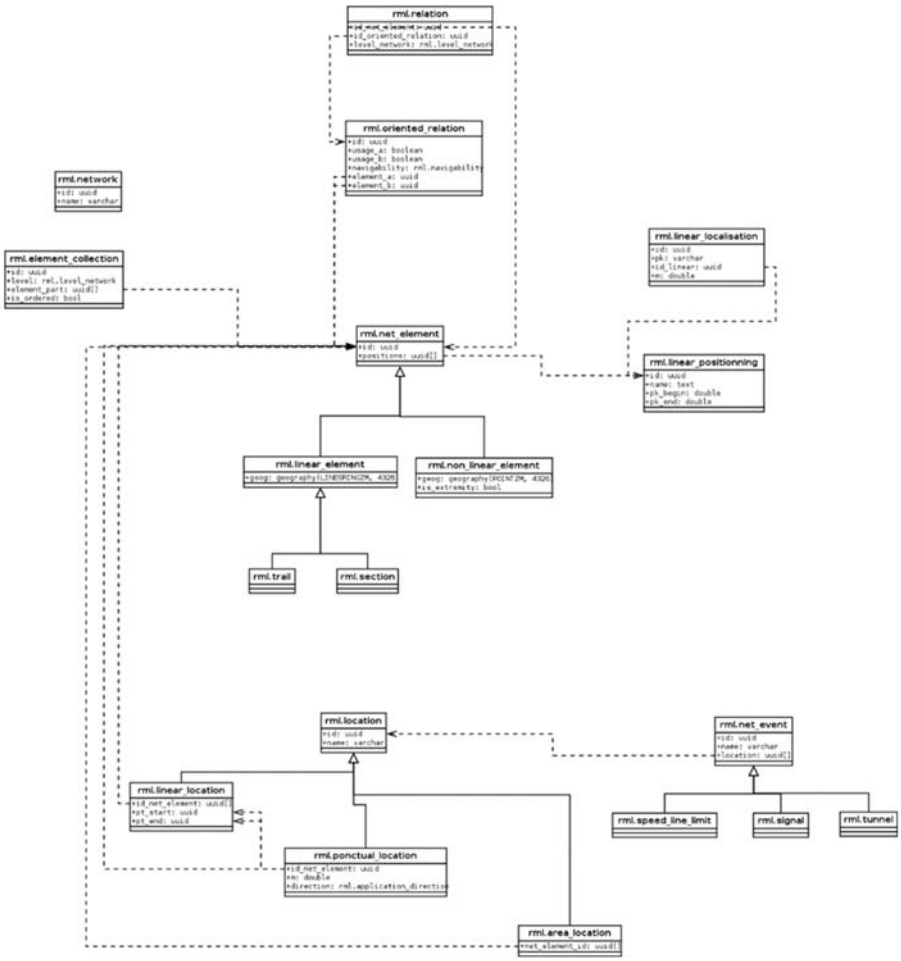
Historically each railway function has developed its own method of representing the railway network appropriate for its own purposes. Merging these different views into a consistent form that supports intelligent cross-function analysis has always been challenging.

Network Rail have advised that RailML 3, presently under development, will be used as the interchange medium for infrastructure data. The key difference between RailML 3 and the current RailML 2 is that version 3 will be synchronised with the UIC RailTopoModel [18] which is a conceptual model of a rail network at the physical and operational levels intended to harmonise the interchange of infrastructure data for specific important purposes across Europe.

0012 The RCM data architecture must be compatible with a rail network model based on RailTopoModel.

A database schema diagram for this model is shown in Figure 16.

Figure 16 - RailTopoModel database schema



The RailTopoModel model supports views of the railway at any level of detail. In particular it considers the physical network (rail sections, switches and crossings, buffer stops) and three standard abstractions of it at different 'zoom levels' called Micro, Meso and Macro.

The Micro level is at the track and switch level and is used to model physical routing possibilities of trains between and along tracks.

The Meso level merges groups of tracks and switches into either Operational Points or Trails and so is equivalent to the timing and routing model using TIPLOCs and Running Lines.

The Macro level merges groups of Operational Points and Trails into a higher route-level view, equivalent to the passenger journey planning view of routes between stations.

RailTopoModel defines processes for working out the Meso and Macro levels from the Micro level via 'Aggregation'.

The model also allows other levels to be defined for business-specific purposes.

Positioning on the network

RailTopoModel supports Linear Positioning (such as the ELR / mileage scheme described in Section Track positions) and Geographic Positioning (latitude, longitude, height or, more generically, x, y z: see Section Locations and places). It has its own translation mechanism using the concept of an Intrinsic Position which is the proportion of the length of a track section, ranging from 0 at its start to 1 at its end, with the ability to map between this position and any of the Linear Positioning schemes defined.

0013 The RailTopoModel intrinsic measurement system must be used to support mapping between different local positioning systems (e.g. the Network Rail and LUL metrics).

Identification of network sections and nodes

RailTopoModel considers every node (=switch, crossing or buffer stop) and track section to be a NetworkResource which is uniquely identified by a UUID and which may also be named.

0014 Every switch / crossing / buffer stop and every track section must be identified by a UUID.

Fixed railway assets and the network

This section defines the conceptual model behind the physical railway network and the operational and commercial views that may be overlaid upon it. This model underpins the way in which sections of the network are represented, assets associated with the network are identified, and business processes associated with the use of the network and its capability are structured.

The model comprises 3 topics:

- The structure of the network and the layered set of views that can be taken of it
- Fixed single assets positioned on or near the network
- Linear assets which make up the network itself

Network Rail are the custodians of the model of the network. A new integrated view of the network is being designed as part of the ORBIS programme – this is the Rail Infrastructure Network Model: Fixed Point Segments and Assets.

[This section should be seen as provisional. Standards will emerge from the Network Rail ORBIS programme for the identification and location of fixed assets.]

Fixed point segments and assets

Fixed point segments and associated assets are those such as switches / crossings / signals, and OLE masts, with a single point location.

0015 Each such segment or asset must be able to be associated with a single geographic position.

0016 Each such segment or asset must be able to be associated with a Linear Positioning position on the railway network, together with a transverse offset, conventionally taken from the track bed centre line, with negative distances to the left and positive to the right of the line when looking in the direction of increasing linear distance.

0017 Each such segment or asset must be associated with a known segment or asset type.

0018 Each such segment or asset must be identified by a UUID or a unique identifier from which a UUID can be found from a recognised registry.

Fixed linear segments and assets

[This section should be seen as provisional. Standards will emerge from the Network Rail ORBIS programme for the identification and location of linear assets.]

Fixed linear segments and assets are items such as sections of track, foundation, OLE, cable, tunnel, and pipe, which stretch from one location to another along the track.

The characteristics of this type of segment or asset are:

- The hierarchical relationship of segments is not as clear-cut as it is for point assets. For track as an example, whilst it is possible to see a clear segment existing for the whole length of the link between adjacent nodal points, there may be reasons to split the track into shorter segments – such as to match with the structure of OLE or formation, or to share boundaries with speed limit changes

- There is no straightforward mapping between the segments and the assets that fulfil them. For example, sections of rail may be longer than a short segment; or may be much shorter if they have been repaired or partially replaced
- Both segments and assets will be identified by a start and end track location. The format of these references will be defined by Network Rail

0019 Sections of linear assets must be able to be identified with a pair of track locations on the same section of running line.

0020 References to linear assets must conform to Network Rail standards.

Moving assets

Vehicles

The fundamental element of rolling stock assets is the Rail Vehicle. Each vehicle is identified by a European Vehicle Number (EVN) which is unique within Europe and which has a standard encoding which includes the vehicle's owner and a check digit.

0021 The data architecture must support the identification of vehicles by their EVN.

Railway Group Standard GM/RT2453 mandates additional information which must be recorded for identification of rail vehicles.

0022 The data architecture must support all the data items required by GM/RT2453.

The Rail Industry Standard RIS-2706-RST recommends a list of other data items that rolling stock related IT systems should support.

0023 The data architecture should support all the additional data items recommended by RIS-2706-RST.

It is important that it is possible to distinguish one end of a rail vehicle from the other, so that components such as wheelsets can be correctly identified.

0024 The data architecture must support the clear identification of the ends of rail vehicles.

Formations and physical trains

Vehicles are assembled together in more or less permanent groupings to provide the rolling stock to operate trains.

Passenger vehicles may be semi-permanently coupled together to form a unit (for diesel or electric multiple units) or a rake (for passenger coaches). A rake may be further semi-permanently coupled to one or more locomotives and/or a driving van trailer to make up a formation.

Any given passenger train may be formed of one or more rakes or one or more units.

Rolling stock may be added to a passenger train or removed from it en route; and the attaching or detaching portion may constitute another train such that trains may be considered to join or split.

Freight trains are composed of 1 or more locomotives and a number of wagons. They may be semi-permanently coupled in the same formation or may be assembled for a single train operation then uncoupled and re-arranged with other wagons for other trains.

Locomotives may run on their own without any wagons.

Within any train formation, it is important to know the orientation of every vehicle so that components can be identified. Some types of rolling stock can only be connected one way in a formation (for example: the end cars of multiple units, HST locomotives, driving van trailers).

When a formation is resourcing a train, the entire formation may be travelling in a forward or a reverse direction.

0025 The data architecture must be able to reflect semi-permanent vehicle formations such as multiple units or rakes of coaches.

0026 The data architecture must be able to reflect transient vehicle formations (whether of multiple units, rakes or individual vehicles) coupled together to form an operational train.

0027 The data architecture must support changes to train formation along the journey of a train.

0028 The data architecture must be able to track the orientation of every vehicle within a formation.

0029 The data architecture must be able to track the direction of travel of the formation on any train.

Operational trains

The railway timetable specifies train movements that are intended to occur on a regular or ad-hoc basis. On any given train movement occasion, rolling stock are provided to operate the movement.

Timetabled train movements are conventionally identified in a number of different ways by different railway management disciplines. The commercial view of a train is different from the operational one, particularly where trains join or split en-route. (In the case of a splitting train, the commercial view is of 2 'sub-trains' coupled together, which part company at the splitting location, each proceeding to its different direction from the same origin; the operational view is of a 'main train' that goes from the origin to one of the destinations, and a 'splitting train' which goes from the point of splitting to the other destination).

Commercially, trains are identified by a Retail Service ID; operationally, by a train describer number (sometimes called a headcode). The headcode does not uniquely identify a train on a day – there can be several even on the same route with the same headcode during the day. Operational computer systems use a number of different methods to generate a unique on-the-day train ID, usually by concatenating the headcode with elements of the train's origin location and starting time.

Movements can be for passenger usage or to move empty rolling stock to where it is next needed.

The train service that operates on any given day is not always the planned service. Changes are made to accommodate incidents that occur, special events or planned maintenance work,

The rolling stock (and train crew) associated with trains are planned ahead of time, the plan being adjusted on the day to accommodate events that occur. To deal optimally with RCM-related events on rolling stock, it can be important to understand what the planned future work of the rolling stock is on the day.

0030 The data architecture must be able to reference and reflect the operational and commercial views of a train, including correct identification of the train in all the circumstances listed.

0031 The data architecture should be able to reflect the future operational and commercial work plan for any item of rolling stock on a train, at least up to the end of the current working day.

Components and compositions

Rail vehicles are assembled from components which may themselves be made of components – for example, a rail vehicle may have two bogies, each of which has two axles, each of which has four bearings. Some of the components have an important role in the safety and reliability of railway operation. Components can be removed from one rail vehicle and fitted to another.

It is important to be able to track the lifecycle of the component assets making up rail vehicles as they move from vehicle to vehicle: this is the job of IT systems such as Component Tracking and the Rolling Stock Register.

Several Railway Group Standards mandate the components of rail vehicles to be tracked and how they should be marked and identified (such s GM/RT 2466 Wheelsets).

0032 The data architecture must support an arbitrarily complex hierarchy of component composition of rail vehicles.

0033 The data architecture must support the requirements for vehicle and component identification and tracing set out in Railway Group Standards.

0034 The data architecture must support the unique identification of assets and components by UUID so that they can be traced over their entire lifecycle.

Domain data model - time and event data

RCM data are intrinsically time- or event-related. Every data transfer must specify unambiguously the event or time period to which it relates, both in terms of the absolute timing or sequence of the datagram and the duration of time for which it contains data.

Events may be rail-related or external.

Rail events

Rail events can come from any of the rail subsystems or their interactions with each other. Some infrastructure-related examples are:

- Operation of a set of points
- Change in aspect of a signal
- Opening or closing of a set of level-crossing gates.

Examples of rolling stock events are:

- Activation of a cab
- Activation or override of any safety system
- Start or shutdown of engine
- Reset of OTMR or other on-train equipment
- Change in traction or brake notch
- Change in door state: commanded to open; open, commanded to close; closed
- Pantograph raise or lower
- Operation of horn

Examples of train/infrastructure interaction events are:

- Passage of a train past a fixed point on the network such as track circuit boundary, trackside detector, OLE neutral section, tunnel, station platform, switch or crossing
- Train timing event, operational or passenger-relevant, including departure or arrival from station or passing of intermediate timing point
- Operation of a train between two timing points

0035 The data architecture must support all types of rail event.

0036 The data architecture must maintain a repository of definitions and codings for the types of rail event recorded.

External events

RCM data may be recorded associated with non-railway events such as road vehicles crossing bridges or level crossings, weather or tidal events, power supply exceedences or outages, trespass or incursion.

0037 The data architecture must support RCM data associated with non-railway events.

0038 The data architecture should support a standard coding or identification scheme for non-railway events.

Event recording

So that events can be correctly timed and sequenced in later analysis, associated with each railway event must be at least one of:

- A locally-recorded timestamp. This should be the timestamp as reported by the equipment detecting the event. Given the limitations of logging

equipment, this may not correspond to standard UTC because of clock drift or the effect of equipment restart, but it should increase monotonically for events recorded between restarts

- A sequence number which must be unique and monotonically increasing for a given sensor or rail asset
- A correct or best-estimate UTC timestamp. This may need to be calculated later based on derived equipment clock offsets or known contextual information such as timings for the same event recorded elsewhere

The timestamp should be sufficiently precise to prevent any possibility of confusion with another railway event of the same type at the same location.

The architecture must define a standard list of event types covering all events of cross-industry interest.

0039 Data architecture users must categorise each event-related datagram with one of the standard event types.

0040 Data architecture users may create their own event types to add additional detail to the standard event type.

0041 The data architecture must support the time-stamping and sequencing of events.

0042 The data architecture must support the use of as-recorded and best-estimate timestamps.

Dated and timed data

RCM data within a datagram cover a period of time. This may be related to the time of an event, such as for 3s before and 7s after the event; or it may be a known period of actual time, such as from 04:00:00 to 09:00:00 on 13 September 2013.

0043 The data architecture should support the recording of the time range associated with the data in any transfer. This may be relative to the timing of the event.

0044 The data architecture must record times, offsets and durations of data in accordance with ISO 8601.

The semantic web and ontologies

Introduction

For several years, academics and specialists have been making the case for the use of ontologies as enablers of process and data integration in the rail industry. This case rests on the role of ontologies as facilitators of the goals of the Semantic Web [19], in which web technologies enable data to be made available, shared, understood, acted upon and enhanced using a suite of standards.

Figure 17 - Semantic Web Technology Stack (T B-L 2000)

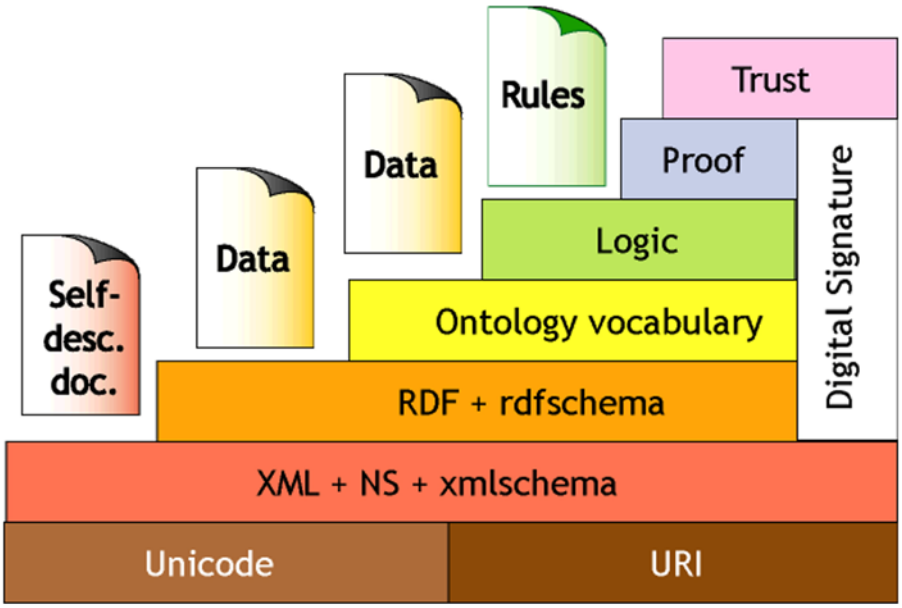


Figure 17 shows the key technologies identified as being required to support the concept of the Semantic Web. In terms of this figure:

- The bottom layer (Unicode and URI) are well-defined technologies in common use now as ways to identify resources and encode data about them.
- The XML/namespace/XML schema layer represents the realisation of data models that support verifiable data transfer. The sections above on data models are set at this layer.

- The next 2 layers – RDF and Ontology – are the subject of this section. These technologies enable the capture and sharing of semantic knowledge – the meanings of data relationships and concepts. They are defined standards which are gradually coming into use.
- Higher layers of this stack do not yet have standard technologies to represent them so sit more in the academic than industrial domain. However, they are likely to become important in future.

The development of these web-based standards and tools for creating, managing and applying ontologies has opened up possibilities for applying the approach in real data integration scenarios.

An ontology adds a layer of meaning on top of a shared data model, because it is capable of representing the concepts implicit in the data model and the relationships between them. This means it can address 'semantic heterogeneity' – different interpretations of concepts – between different IT systems, on top of the 'structural heterogeneity' – different data structures – addressed by a shared vocabulary / data model. This simplifies the process of integrating these IT systems.

An ontology also allows relationships of meaning to be defined between different data representations, allowing the ability to 'reason': to draw conclusions about data items which are implied by the semantic relationships contained in the ontology, without them being explicitly stated. This enables a new class of software tools – "reasoners" - to be developed, which can interrogate the ontology to drill into these relationships or chain them together.

The data architecture supports the following uses for the ontological approach:

- As a means of capturing concepts associated with railway assets and data about them
- As a means of systematising, checking and automating the mapping between the different data model layers described in the sections Conceptual data models to Domain data model - Time and event data.
- As a means of capturing and representing data about individual assets and components
- Supporting the presentation of existing industry reference data in ontology-friendly ways which facilitate the application of novel software approaches such as automated reasoning

- Supporting the management and development of shared ontology repositories

These uses are described in the following paragraphs.

Expressing asset data concepts in an ontological way

An ontology enables the concepts used in railway asset management and operation to be captured in a formal way, represented unambiguously and used by software to interrogate and understand the relationships between concepts and to derive implications from them even if not explicitly stated. (The classic simple example for this uses logic such as: Facts: 1) Unit 319 012 is a train; 2) Unit 319 012 has a pantograph. Conceptual rule: If a train has a pantograph it is an electric train. Inference: Unit 319 012 is an electric train.) More sophisticated applications might include:

- Understanding the detailed structure of complex assets. This supports, for example, deriving health assessments for a composite asset (such as an operational train) made up of a collection of simpler assets (such as a train made up of two units, each comprising 4 vehicles, each with 2 bogies, each with 2 axles, each with 2 bearings, each of which we have a condition rating for)
- Bridging operational and engineering domains. This supports, for example, calculating the impact in passenger lateness minutes for a morning peak for a section of route based on the inability to operate a single switch and thus a reduction in the number of possible routes through an interlocking; or the calculation of the expected usage of a single section of OLE based on the future train timetable
- Representing fault trees and failure modes in a shareable and extensible way, including with fuzzy logic. This will enhance the ability to build up best-practice fault-finding and diagnostic algorithms and link asset state to likely future life

These concepts can be represented in data stores using the RDF, RDF Schema and OWL technologies. They can be made available for use by publishing the data stores in standard ways such as web services or SPARQL endpoints.

Figure 18 - Mapping service powered by shared ontology

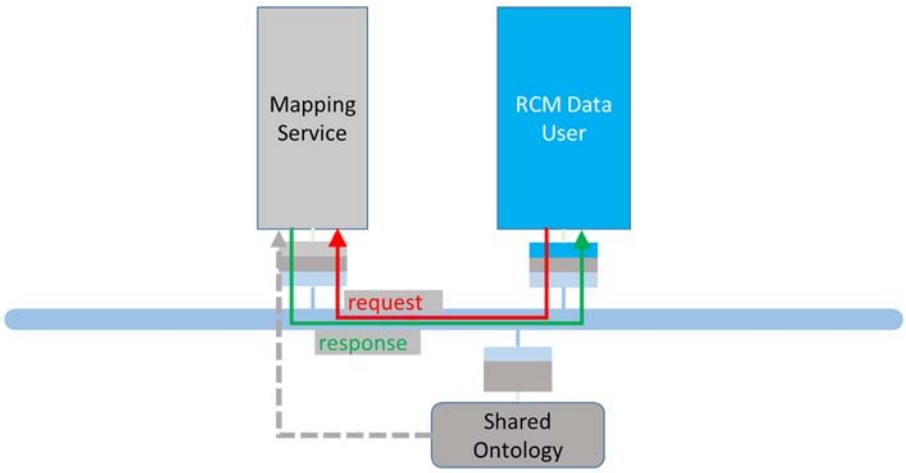


Figure 18 shows the style of application that can be supported by the use of this ontology-based view of the data concepts: a software service which provides a mapping function – say from an engineering-based view of the rail network to an operationally-based view – using the shared ontology to inform the mapping. The benefit of the use of ontologies here is in the in-built extensibility of the approach: changes to the network data, to the level of abstraction of the network views concerned or of the exact outputs required can be accommodated with a minimum of change to software.

The technologies mentioned have the following functions and characteristics:

Resource description framework (RDF) is a simple method for expressing facts about concepts or objects using a simple 'triple' arrangement of subject-predicate- object (such as: Wheelset IsPartOf Bogie, where Wheelset would be the subject, Bogie the object and IsPartOf the predicate describing the linkage or relationship). Formally, the subject, predicate and object are URIs – they are unique. The language does not constrain subjects, objects or predicates in any way.

SPARQL is a query language for data expressed in RDF triples, similar in concept to SQL used to query relational databases. An RDF data source queryable using SparQL is called a SparQL Endpoint – it is usually represented as a URL web address. The result of a SPARQL query of an a result set presented in any of a number of standard formats (XML, JSON, CSV, TSV) or a TRUE/FALSE result – such as does any item matching the query exist.

RDF Schema is a way of adding some formality and standardisation to the types of object, enabling types of objects and concepts ('Classes') and descriptive attributes of them ('Properties') to be defined, as well as subclasses ('subClassOf' predicate) and 'is a' relationships ('type' predicate), subproperties ('subPropertyOf'), cross-references ('seeAlso'), formalisation of vocabularies ('isDefinedBy') and documentation ('comment').

OWL is the Web Ontology Language. It includes all the ideas in RDF and RDF Schema, and adds to them some key features such as the ability to define classes of objects by rules on other objects (for example: that a train set with a pantograph or with a 3rd-rail pickup is an electric train set); to combine these rules using the ideas of set theory (such as UNION, or INTERSECTION) or inverse logic; to have rules based on the number of objects in a class; to verify the correctness or validity of the ontology by detecting contradictions (where the rules defined are logically inconsistent) or unsatisfiable classes (where the rules mean that no actual individual object could be in the class).

Fully-featured OWL 1 (called OWL-Full) enables a very wide range of ontological concepts to be represented. However, it has some drawbacks relating to how practicable it is in operation, because it is 'undecidable' – no software reasoner will be able to provide an answer to all the possible questions that could be asked of the ontology. For this reason, subsets of OWL were defined which offered nearly all the features of OWL-Full but had some constraints to enable them to be decidable and to perform well for common types of usage:

- OWL-DL ('Description Logic') is decidable, at the expense of removing the ability to treat classes as things and to redefine elements of the OWL language. These constraints are not normally seen to be too onerous
- OWL-Lite is a subset of OWL-DL aimed at simplifying the design of ontology modelling tools. In practice the constraints introduced to enable this have been found a) to be onerous and to restrict modelling capability; b) to be able to be circumvented by using other rules in the subset. It is not much used

Given the likely usage of the ontology in this context, it is important that a) it is decidable – there is a deterministic result for any query; b) it is expressive because of the range of types of concept that need to be represented. For these reasons the shared ontology should conform to OWL-DL.

There is a new version of OWL, OWL 2. Any ontology which is legal in OWL 1 is still so in OWL 2, so the OWL-DL subset will continue to be useful. However,

there are new subsets ('profiles') of OWL-2 which have been defined to make it useful for specific purposes:

- OWL 2 QL. A subset which is optimised for performance when dealing with ontologies whose data is mostly stored in SQL databases
- OWL 2 EL. A subset which is optimised for performance when dealing with sprawling ontologies with many complicated numerical or range-based classifiers
- OWL 2 RL. A subset which is optimised for handling rule sets such as might be imported from an existing rules engine

In the context of rail RCM data, nearly all asset and RCM data items reside at some point in SQL databases and need to be queried from them. This suggests that the OWL 2 QL profile represents a suitable subset.

0045 The data architecture must support a shared ontology repository.

0046 The data architecture must support the use of OWL / RDF data store to represent a shared ontology.

0047 The shared ontology should be based on the OWL-DL sublanguage of OWL 1 or the OWL 2 QL profile of OWL 2.

0048 The data architecture must support the exchange of RDF data in standard RDF formats using any of the well-known serialisations: RDF/XML, Turtle or JSON-LD

0049 The data architecture must support the use of SPARQL Endpoints as a form of data adapter.

0050 The data architecture must support the transfer of SPARQL result sets in any of the standard forms defined in SPARQL 1.1, including XML, JSON, CSV and TSV.

Guiding data modelling

A data model comprises entities and relationships between them. The business of identifying and describing the entities and relationships in the model is a human-driven exercise that parallels the one that would be necessary to build an ontology of the same things.

Ontologies can support and simplify the data modelling process in these ways:

- Many important domains that overlap with the rail RCM data domain have already been modelled and are available as standard ontologies which are in common use. Examples are FOAF (Friend of a Friend) [20] for people and relationships, Dublin Core [9] for documents and their attributes, QUDT for engineering units and quantities [21] and PROV [8] for data provenance /

processing information. This means that the conceptual model for these types of data is already widely understood, thus avoiding the need to 're-invent the wheel'

- The use of an ontology will act as a guide to the modellers responsible for extending and enhancing the Data Interchange data models and updating the Domain data models, helping to ensure that such extensions are done in a consistent way
- Ontologies can be used to derive the mappings between the data structures of connected IT systems and the data bus Data Interchange data models (as shown in Figure 4)
- Although not currently well-supported with toolkits, there is a realistic prospect that in the near future it will be possible to automate the generation and maintenance of data models based on the shared ontology. This will improve the agility and adaptability of the data architecture

0051 The data architecture should use existing well-known ontologies where possible. Specifically, FOAF, QUDT, Dublin Core and PROV should be used.

0052 The data architecture should support mappings between the shared ontology and the conceptual, domain and data interchange data models.

0053 The data architecture should support the use of the shared ontology to facilitate the creation of system to data bus mappings in Data Adapters.

Expressing asset data in an ontological way

As well as the use in managing rail data concepts and entities described in Section Expressing asset data concepts in an ontological way, an ontological approach can also be taken to data describing actual rail assets and RCM data. This can be used in the following ways:

- To classify assets – to allocate them to a recognised item type from the shared ontology. This then means that all the rules and inferences defined in the ontology for the class of asset are then applicable to specific assets
- To relate assets to each other. Describing the relationships between, say, nested components of a rail vehicle (such as vehicle -> bogie -> axle position -> side -> bearing position -> bearing) in an ontological way ('isPartOf' predicate, for example) can simplify enquiries and deductions about the whole asset based on the status of its components

- To link data about assets from different systems and sources. An ontological view of mappings across naming schemes (such as linking locations coded by STANOX and NALCO, or parts by design type and manufacturer's product code) simplifies the task of joining data about the same asset held in different systems with different codings and concepts. ('sameAs' predicate, OWL's 'functional properties' and 'inverse functional properties' which enable sameness to be inferred)
- To achieve these uses of ontologies, it must be possible for information about assets to be presented to data architecture users in a standard ontological format. This means expressing them as RDF triples conforming to the shared ontology. Since the asset data are known to connected reference systems (per item 4 of Figure 4), this means that the data adapters for these asset data types should include the ability to provide data in one of the standard ways of sharing RDF triples

0054 The data architecture must support the use of RDF triples to represent data about assets and their relationships to each other.

0055 The data architecture should support the creation of data adapters which map existing reference data sources into RDF triples consistent with the shared ontology.

Using the shared ontology

The shared ontology should itself be available to users of the data architecture, with its own data adapter as shown in Figure 4. This will enable creators of mapping services and data adapters to refer to the ontology whenever they need to.

There are a number of standard ways of expressing and exchanging OWL ontologies (called 'serialisations'), each optimised for a specific purpose but all equivalent in terms of the ontology itself. The most useful are:

- RDF/XML. This expresses the ontology as RDF triples encoded in XML. The advantage of this serialisation is that it is supported by all ontology tools. It is hard for humans to read, however
- Manchester Syntax. This is a human-readable way of expressing an ontology
- OWL/XML. This is an XML representation of the ontology using an OWL-specific schema
- Turtle. This is a human-readable and concise representation

0056 The data architecture must make the shared ontology available as a download or via a web service using a standard serialisation such as OWL/XML. RDF/XML or Turtle.

Data formats

Basic data formats

At the lowest level, the architecture requires standard data items to be expressed in the specified form. Where a form is not specified, data suppliers are free to use existing formats or choose new ones.

Standard data items for which formats are defined are:

- Dates and times – using ISO 8601. This standard reduces the confusion that exists with date formats and time zones
- Geographic locations – using ISO 6709. This means that locations expressed in terms of latitudes and longitudes or OS grid references are done in a consistent way using an understood frame of reference
- Fixed and floating point numeric values expressed as text – using the standard text representations defined for XML Schema. These are common-sense representations which can be used without error in other expressions such as comma separated values files
- Units of distance, mass, force. These must conform to the SI standard as set out in [22], except where railway-specific measures such as miles and chains are in use, where Network Rail standards should apply

Base information types - representation

This section contains definitions of basic data types which are used in many types of data interchange. Standardising on these definitions will improve the quality of data interchange and reduce development or maintenance costs, even if no other architectural initiatives are followed up.

All the definitions here use international or industry standards, thus improving the ability to share information with that from other non-rail domains without confusion.

Dates, times and intervals

0057 Dates and times must be expressed in a form compatible with ISO 8601 [23].

Dates

For dates, this means the standard form YYYYMMDD or YYYY-MM-DD. Years must always be expressed with 4 digits. The date may be truncated to refer to a whole month (in which case only the form YYYY-MM is allowed, not YYYYMM) or a whole year.

Times

For times, the standard representation is HH:MM:SS or HHMMSS. Truncation is allowed to HH:MM or HHMM or HH. Decimal fractions of the least significant part (usually seconds) can be added to arbitrary precision as HH:MM:SS.dddd.

Time zones and changes between summer and winter time can cause confusion, particularly when referring to times read from field sensors which may not automatically adjust their clocks when summer time starts or ends. To avoid this:

0058 Times must always either be expressed in Universal Coordinated Time (UTC) (roughly equivalent in the UK to Greenwich Mean Time), which is indicated in an ISO 8601-compliant time by the indication Z (HH:MM:SSZ); or have an explicit offset from UTC specified. For example, a time in Greenwich Mean Time would be expressed as HH:MM:SS+00:00 or HH:MM:SS+00; in British Summer Time would be HH:MM:SS+01:00 or HH:MM:SS+01.

Timestamps

0059 Combined dates and times (timestamps) must be represented as a Date element (as above), a T separator and a Time element – i.e. similar to YYYY-MM-DDTHH:MM:SSZ or YYYYMMDDTHHMMSSZ.

Clock adjustments

Where a date or time has been taken from an unsynchronised clock and therefore may differ from the actual clock time, this must be indicated in the name of the field containing the date and time.

For such times, it may be possible to calculate an adjustment to the time to get to a clock time. Such an adjustment or offset must be clearly identified in its field name and must be expressed in the “Duration” format allowed by ISO 8601.

0060 The data architecture should use a standard nomenclature for times and dates to distinguish between ‘as-recorded’ and ‘known good’

values.

0061 The data architecture must support the use of time offsets to correct for clock drift.

0062 The data architecture should use a standard nomenclature to identify time offsets.

0063 The data architecture must show time offsets in the format of ISO 8601 durations.

Locations and places

All fixed railway assets have locations which can be expressed in geographic terms (latitude and longitude or OS grid reference) or as a railway position (typically Engineers' Line Reference (ELR), milepost distance expressed as miles + chains or miles + yards, track ID).

Geographic locations

0064 The data architecture must support the use of geographical location to refer to an asset's position and to locate other assets nearby or associated with it.

0065 Geographic locations (expressed in earth-centred co-ordinates) must be expressed using the form defined in ISO 6709 in the data architecture.

This standard allows any properly-registered geographic reference system to be used. In particular, it allows GPS latitude / longitudes using the WGS-84 spheroid and Ordnance Survey grid references to be used.

Unique identifiers

Every asset should have at least one unique identifier. There may be a well-known 'real world key' such as an EVN for a rail vehicle: the architecture must enable this to be used.

To guarantee uniqueness of assets, a genuinely-unique identifier must also be used for every entity. The standard type of identifier used in this context is the UUID, whose format and possible methods of derivation are described in RFC 4122 [16].

A UUID is basically a 128-bit binary number which can be generated by a number of algorithms on demand with a very high probability of uniqueness. The data architecture does not need to generate UUIDs for all entities – it is perfectly permissible for it to use UUIDs generated elsewhere such as by asset

owners or maintainers – but it may provide a UUID service to assist data interchangers to acquire UUIDs on demand where needed.

Some methods of generating UUIDs would generate consecutive UUIDs which look very similar to each other, differing in only a few digits. These represent a potential security risk, as they enable readers of a UUID to attempt to simulate another one for a different resource. Such methods should not be used.

In the ontological languages RDF and OWL, items are identified by URIs which are unique in web space. It is perfectly permissible to use UUIDs as URIs if they are correctly formatted as required by RFC 4122.

0066 Wherever a segment, asset or any other item referenced by the RCM data architecture is identified by a unique ID, this must be a 128-bit UUID as defined in RFC 4122 [16].

0067 Where UUIDs are represented in strings, such as in URIs, they must be shown in the format described in RFC 4122 – similar to 'urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6'.

0068 UUIDs should be hashed so that sequentially-issued UUIDs bear no obvious resemblance to each other, to prevent spoofing of UUIDs

0069 The data architecture should support industry-wide services for looking up the items referenced by existing UUIDs and for issuing new ones.

0070 The data architecture may offer its own service for providing UUIDs on request to support the use of MIMOSA concepts.

Railway data types – infrastructure

Railway segments and assets shall be identified by standard methods. (See the People and processors section for a discussion of the meanings of the terms Segment and Asset). Where there is likely to be confusion or where different reference schemes may refer to the same segment or asset, the use of globally unique identifiers (UUIDs) should be strongly considered, as defined in MIMOSA SDAIR [17].

- Fixed point segments and the assets occupying them (for example, items such as signals, cabinets, OLE masts) must be identified clearly by unique identifiers issued by the network manager
- Locations on the railway network ('Track Locations') must be identified by route position, track identifier (where appropriate) using one of the standard format(s) set out by Network Rail

- Linear segments or assets (such as sections of track, foundation or OLE) must be identified by asset type and a pair of Track Locations (from and to)

0071 The data architecture must support any standard mechanism for identifying railway fixed or moving assets using the unique identifier appropriate to the function.

0072 The data architecture must require a UUID to be supplied for any asset.

Locations and topologies

Track positions

[The contents of this section are provisional pending Network Rail's defining a standard representation for a track position.]

In this representation:

- ELR must be a valid Engineers' Line Reference (3 or 4-char alphanumeric string)
- Mileage must be expressed in either Imperial units (miles and part miles) or Metric units (km and part km)
- In the Imperial form, mileage must be shown as miles and chains in the form mmm:cc or miles and yards in the form mmm+yyyy. The miles figure mmm need not be zero-padded; the chains cc or yards yyyy must be padded to the left with zeros to a total length of 2 or 4 digits respectively. cc must be in the range 00 to 79; yyyy must be in the range 0000 to 1999. (Railway miles may be more than 1760 yards long because of milepost positioning error)
- In the Metric form, 'mileage' must be shown as km and fractional km in the form kkkk.mmm
- Track ID must be a valid numeric track ID using standard nomenclature such as 2100 or 1200 or well-known Sectional Appendix identifiers such as DF = Down Fast

0073 Positions on the railway network must be identified by one of the standard means defined by Network Rail. These will include ELR, Mileage, Track ID.

0074 Track positions must be able to be represented as Miles:chains, Miles+yards or km.mmm.

0075 Running Lines must be identified using one of the standard naming mechanisms. The field name should indicate which one (such as RunningLineGEOGIS)

0076 The data architecture should recommend naming conventions for field names to indicate which naming mechanism they refer to.

0077 The data architecture must require any running line to have a UUID to support mapping between different representations of the same running line.

Topological locations

[This section should be seen as provisional. Network Rail may in due course specify standard representations of railway locations].

[Network Rail has expressed the view that it will in future use the RailML standard for railway data interchange. This has an 'intrinsic' method of identifying track location which is distinct from any specific railway scheme but allows mapping between different ones on the same track.]

Named locations on the railway network may be identified by any of the standard location codings used in the UK railway. The coding in use must be identifiable from the field name. Typical codings might be:

- Operational locations, identified by valid TOPS STANOX code. This is a 5-digit integer, padded to the left with zeroes
- Timing locations, identified by valid TIPLOC code. This is a 7-character alphanumeric string
- National Locations, identified by valid NALCO code. This is 4-character alphanumeric string which may contain only numeric digits. If it does only have numeric digits, it must be left-padded with zeroes up to 4 characters in length
- Passenger stations, identified by valid CRS code. This is a 3-character alphabetic string

Mapping between the different location codings is outside the scope of the RCM data architecture. However, we anticipate that it will be necessary to do this mapping, for example when calculating the operational or network capability impact of an infrastructure failure. We have thus assumed that the industry (probably in the shape of Network Rail, via the ORBIS programme) will provide a mapping service which will enable locations in one coding to be expressed in another. We suggest a protocol for this in the section Industry Reference Data Requests.

Topological links

[This section should be seen as provisional. Network Rail may in due course specify standard ways of referring to sections of route.]

0078 Sections of railway route must be identified as running between a number of topological locations, with a directionality indicator and an optional track indicator (for example, KINGSX->FINSBPK or KINGSX->(FL)->PBORO) or by Sectional Appendix route identifier.

The topological locations can be expressed in any of the ways described in the Topological locations section, though both the start and end location must use the same method as each other.

0079 The directionality of topological links must be indicated as being unknown or irrelevant, from first to last location, from last to first location or both directions.

Table 11 - Directionality indicators

Indicator	Means
-	Directionality not known or not relevant
<>	Both directions
->	Direction from first location to last location only
<-	Direction from last location to first location only

0080 The Track Indicator must be either a Track ID in GEOGIS format or an operational line identifier as shown in the Network Rail Sectional Appendix for the route concerned.

Mapping between different route-level representations is outside the scope of the RCM data architecture. However, it will be necessary to be able to do this to support some RCM-related business processes such as calculating the network-level impact of the failure of an asset on a specific track section. We have assumed that this mapping will be able to be done by means of a mapping service, probably provided by Network Rail as part of the ORBIS programme. We have suggested a method for doing this in the section Reference Data Store.

0081 The data architecture must support external services which map between different route-level representations of the railway network.

Shared reference data

Introduction

The sharing of data across the rail industry requires a set of shared concepts and data models as described in Section Data models. It also requires a shared understanding of the data items themselves that populate the data models – a shared taxonomy and shared vocabulary.

Given the volume and rate of change of UK rail assets and components and the fact that certain industry systems are designated as the master data source for each type of asset, it is neither desirable nor realistic for there to be a shared data store for them all as part of the data architecture. We have therefore assumed that the system sponsors for the master reference systems (such as Rolling Stock Library (RSL) or track (ORBIS, ultimately)) will provide lookup functions for assets.

Figure 19 - Request and response reference data lookup

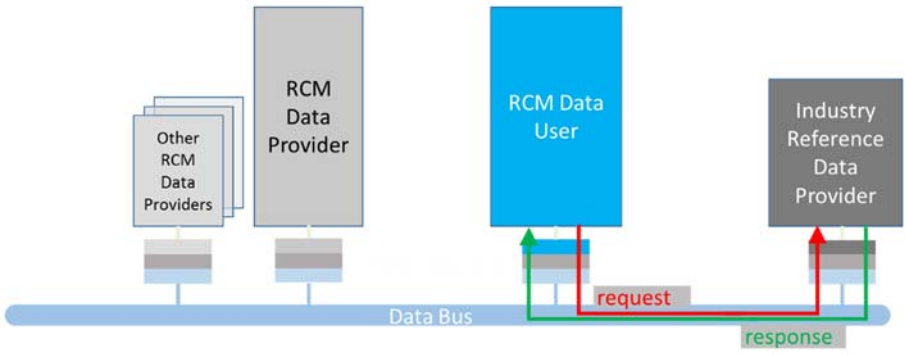


Figure 19 shows a possible arrangement for this type of data lookup based on a shared service.

The reference data providers would be facades built on to existing industry IT systems using the standard data adapter mechanisms already described. The queries and responses would use standard web service protocols. Typical queries might be (of a rolling stock register system): list all the component parts of vehicle with EVN 937034512342; (of a network model): give the track

location nearest to Lat 51.2315N, Long 1.5432W; (of an AVI service): give the full consist of the train that passed the point at mileage 23:46 on ELR ECM1 track 2100 heading northbound at 12:22 on 5 May 2014, for which one RFID tag said 937034512342L.

The owner or sponsor of each reference system would need to provide the web services. The exact format of the data returned by the service would depend on this sponsor, but it should conform to the Shared Data Model defined by this data architecture.

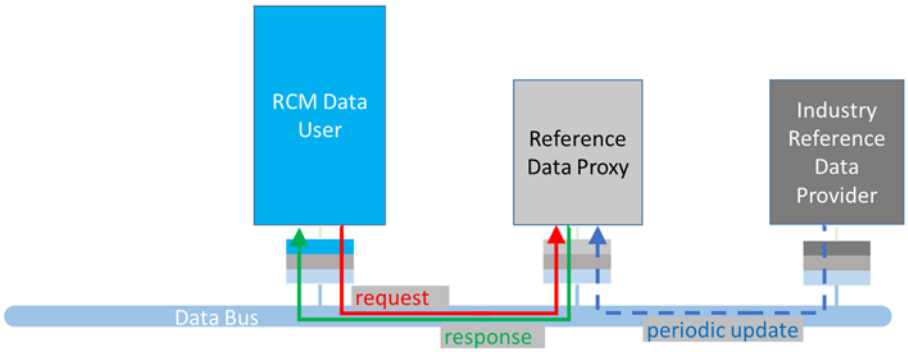
The benefits to the industry of this type of reference data service will be great and will not be limited to the demands of RCM data interchange. They will result in a massive improvement in data currency and quality across the rail network and will enable the sharing of reliable data on a much greater basis than now.

0082 The data architecture must support the use of external service-based lookups to standard industry reference data

Additionally, there may be a case for shared proxies for some or all reference data. These would be centrally-managed servers providing easy-to-use standard reference lookup web services, based on periodic data extracts from the actual reference data sources.

It may be desirable, however, for proxy data stores for this type of information to be managed centrally to reduce the demand on the master systems or to provide the analysis necessary to support more complex data services such as those that need an ontological approach (see Section The semantic web and ontologies) to satisfy them. Figure 20 shows the basic setup. The Reference Data Proxy gets a periodic refresh of the actual reference data from the industry master source; RCM data users do lookups on the proxy rather than the industry reference data source itself.

Figure 20 - Reference data proxy



0083 The data architecture should support the use of Reference Data Proxy services to reduce the load on industry systems and to provide functionally richer lookup services.

Types of reference data

Various types of reference data item or entity have been introduced in the proceeding sections, together with the recognition that they need to be identified by UUID; and that it may be necessary to translate between different representations of an entity.

These reference data fall into these major categories:

- Entities which exist in the railway network at large, for which a reference data source exists already in the industry. For these, the role of the RCM data architecture is not to manage the master data sets which store the reference data, but to act as a broker and reference medium for the UUIDs and their various aliases; and to support lookup functions in the form of web services (as described in Section Metadata).
- Entities which are specific to the management and use of RCM data, such as alarm types, data units, sensor types, health levels and statuses, fault and failure modes, algorithm types. For these, the data architecture will need to support the storage, reference and management of the reference data.
- Entities which define the mappings between abstract and specific data structures – for example which indicate how a recognised railway topology such as the timetabling model is generated as a specific example of a MIMOSA segment hierarchy. This includes the shared ontologies described in the section *The semantic web and ontologies*.

- Rules which define the relationships between different concepts – such as between a vehicle and an electric multiple unit. These may be made available using ontology-based methods such as RDF triples as described in Section The semantic web and ontologies.

These categories are described in the sections below.

The time or history dimension

Reference data typically deal with time in one of 3 ways:

- They ignore it. There is a single reference value and dependent attributes. If a change occurs to a reference data item, this would impact past data which refer to it as well as current and future data.
- They have effective date ranges. For each reference value there is a start date and an end date. This means that history can be kept, there is always a current value, and values can be entered to become valid at a future date. This might apply to unit costs or similar slowly time-varying data.
- They have a specific date on which each entry applies. This would be true of, for example, a train run, which is essentially a different entity for every date.

Each type of reference data needs to be treated in a way appropriate to its usage in the industry.

0084 The data architecture must support the use of time-invariant, effective-dated and specific-dated reference data items

Railway entities

All the entities described in this section have reference data about them already stored in master data systems distributed around the rail industry. The role of the data architecture here is to support the lookup of this data from these master data systems.

Other non-master systems (such as individual rail operators' IT systems which manage rolling stock or operations) may have their own way of referring to the same items, or their own coding systems for faults or incidents

In many instances there are shared code lists and vocabularies which are implicit in the industry but have no formal master system. The data architecture can provide a master reference for these if they are relevant to RCM data interchange.

Fixed railway entities

Valid identifiers and descriptions of all fixed railway entities. They comprise all conceptual items (such as Timing Points and other location codes), point assets with a single location on the network (such as signal cabinets, switches and crossings), linear assets which occupy a section of route (such as track or OLE), and areal assets which occupy a more complex subset of the network (such as a junction or station area).

Moving railway entities

Valid identifiers for all movable railway entities (rolling stock). These comprise vehicles, quasi-permanent formations such as multiple units, transient formations such as units coupled together to operate a train, operational trains.

Reference data store

To support the MIMOSA framework and the derived realisations of it for UK rail asset types, sensor types and data types, a set of reference data tables needs to be populated. These are relatively static tables, changing only when new types of asset template, sensor or data type are required by a project. They are also small tables. The whole data store can therefore be managed as a set of text files.

Standard ways of representing UK rail notions such as track location, train ID, vehicle structure etc will be defined outside of the data architecture, by the sponsors of the systems or by an industry-wide specification document such as a Rail Industry Specification.

Other reference data types exist which are more specific to the RCM data interchange domain. Examples include:

- Agent role types
- Agent types
- Alarm types
- Asset readiness states
- Asset types
- Average types
- Legal MIME types
- Criticality levels

- Data quality types
- Data source types
- Engineering units
- Enterprise types
- GPS datums
- GPS precision levels
- Health level types
- Resource types
- Measurement location types
- Network connection types
- Network types
- Priority level types
- Reference unit types
- Segment types
- Signal processor types
- Site types
- Work item types
- Work management types
- Task types
- These data types should be standardised across the UK rail industry to support a consistent understanding of the RCM data they describe
- The IT infrastructure implied by the need to store these data items is discussed in the *Reference data store* section

0085 The data architecture must support the management of RCM data reference and lookup data of all types required.

0086 The data architecture must provide lookup data services and downloads of all shared reference data in standard formats.

Metadata

Metadata describe data, thus enabling users of the data to apply it in an appropriate way. Lack of a way to communicate the limitations, characteristics and context of RCM data is one of the main limitations to its effective cross-industry supply and use. The data architecture therefore needs to provide some standard methods for supplying this additional information. It recognises the following specific aspects of metadata, but also provides a mechanism for data providers to enhance these with their own metadata items if required:

- Data source. The originating organisation or IT system must be defined. This must be from a standard list of recognised data sources
- Time or data sequence. The creation date of a data item must be specified. In some contexts, exact correct creation time is not known (for example because of sensor clock drift or inaccuracy). In these cases, data items must be given a unique sequence number
- Data owner. The legal owner of the data can be identified. If done, it must be done using either the 'Dublin Core' standard representation [9] or information in PROV format [8]
- Data licensing or usage restrictions. Data providers may restrict the use of the data to specific parties or specific purposes; and there may be license terms associated with the data. These can be specified, again using Dublin Core methods
- Data units. Every data item must have its engineering unit and scaling factor specified. These must be defined using MIMOSA standards and optionally using the QUDT ontology
- Data quality indicators. The data architecture uses the MIMOSA four-level indication of the status of the data (OK / FAILED / UNKNOWN / NOT_USED) and confidence percentage (0 % = no confidence → 100 % = full confidence) which may be used as applicable; other data quality indicators (such as precision, standard deviation) can be added as required.
- Configuration. Any RCM data processing element at whatever level of the ISO 13374 hierarchy will have stored setup parameters which defines how

it behaves. The data architecture provides a standard method for interrogating and supplying these based on MIMOSA principles

Data suppliers may wish to identify themselves as owners or creators of the data or may need to apply licensing or usage conditions to it; data consumers may wish to know its provenance and details of its applicability to their own usage; Railway Group Standards or other legal stipulations may require that the party responsible for data or actions on it are identified.

All of these concerns are handled in the RCM data architecture by the use of metadata – additional data items which describe aspects of the asset and RCM data itself. The XML schemas in which the data interchanges are specified have extension points defined at which these data items can be included.

0086 The data architecture must support the application of metadata items to any data interchange.

0087 The data architecture must enforce the presence of mandatory data items on data interchanges, including data source and time sequence.

0088 Ownership metadata must conform to the Dublin Core standard or PROV methods.

0089 Data licensing and usage restrictions must be stated using Dublin Core methods.

0090 The data architecture must enforce that data engineering units and scaling factors are specified in accordance with MIMOSA requirements and the QUDT ontology.

0091 The data architecture must support the use of data quality indicators in accordance with MIMOSA requirements.

0092 The data architecture must support the use of data quality metrics defined as required by data interchangers.

0093 The data architecture must support the interchange of configuration and explanation data as defined in the MIMOSA standard.

Data interchange

Introduction

In previous sections we have discussed the content, structure and format of rail RCM data that might be exchanged between users of the data architecture. In this section we consider how the data interchanges will take place.

The underlying principle is that of a Data Bus, where data interchanges take place using standard data formats and standard methods, independently of the data formats used by the participants internally.

This section considers:

- Data models used for data interchange
- Data interchange formats – 'datagrams'
- Sample XML schemas and datagrams
- Data interchange methods
- Asset referencing schemes

The architecture needs to enable data to be interchanged using a number of standard formats. Users of the architecture will generally have a choice of methods for any given application. Where possible, data formats will conform to internet standards (such as the use of MIME types [2] for standard file types) for data content.

The data architecture specifies standard data items and record structures for each type of data to be handled. These can then be packaged up in a variety of formats according to the specific needs of the data interchangers. In descending order of preference, acceptable formats will be:

- XML datagrams. XML schemas will be defined for the key data types: these can be used by data creators and data users to verify the conformance of the data to the architecture specification. Detailed / voluminous data will be able to be embedded in the datagram or referenced by standard URI
- JSON objects with named attributes conforming to a schema defined by the architecture

- Existing rail industry-standard text file formats
- Text file formats such as tab-separated values or comma-separated values, with columns named and sequenced as defined by the architecture
- Multimedia files in a MIME standard type (such as images as image/png, audio as audio/wav or audio/flac, video as video/mp4)
- Novel proprietary structured text formats
- Proprietary binary formats

The intention is that the XML datagram formats are sufficiently general and well-defined that the other textual formats (JSON, most regular rail formats, text files) can be mapped to and from them using XSLT transforms or other simple coding methods.

0095 The data architecture must support data interchange using mandated data formats defined in the data interchange schemas.

0096 The data architecture must provide means to validate XML datagrams against shared schemas.

0097 The data architecture must enable data to be encoded as JSON objects compliant with the shared schemas.

0098 The data architecture must enable data to be encoded as standard TSV or CSV text files compliant with the shared schemas.

Data interchange data models

The sample data models in the following paragraphs have been mapped from the conceptual and domain data models described in the *Data models* section, using well-known and standard representations of asset types, locations and usages common in the UK rail industry. They are therefore close in approach to the data types that will be found in existing IT systems that are likely to be connected to the data architecture for the provision of reference data or the interchange of RCM-related data.

It is intended that these data models will represent the types and structures of data that will be transferred using the Data Bus. Their format should make them easy for data providers to generate and data users to interpret.

These models can be expressed directly as XML schemas which can be used to design and validate actual data interchanges using XML. The XML schemas can also be used to design data interchanges using other acceptable data formats such as .csv and .txt. The key to validity here is the ability to transform the data format into XML using an XSLT or other coded method. Some sample

schemas generated from these data models have been included in Appendix 1.

The mapping process used to create these models is an informal one based on our own understanding of the concepts and entities in use in UK rail systems. Given the importance of conforming these mappings to the abstract data models, it will be valuable to consider automating this process. This is an area where ontologies (see Section The semantic web and ontologies) are likely to have much to offer.

Core data models

The Data Interchange models have their own core data structure which is a specialisation of the abstract MIMOSA one described in Section Basic Entities in the Rail RCM Domain. This is shown in Figure 22 (basic data types) and Figure 21 (conceptual model).

Figure 21 - Basic data types data model

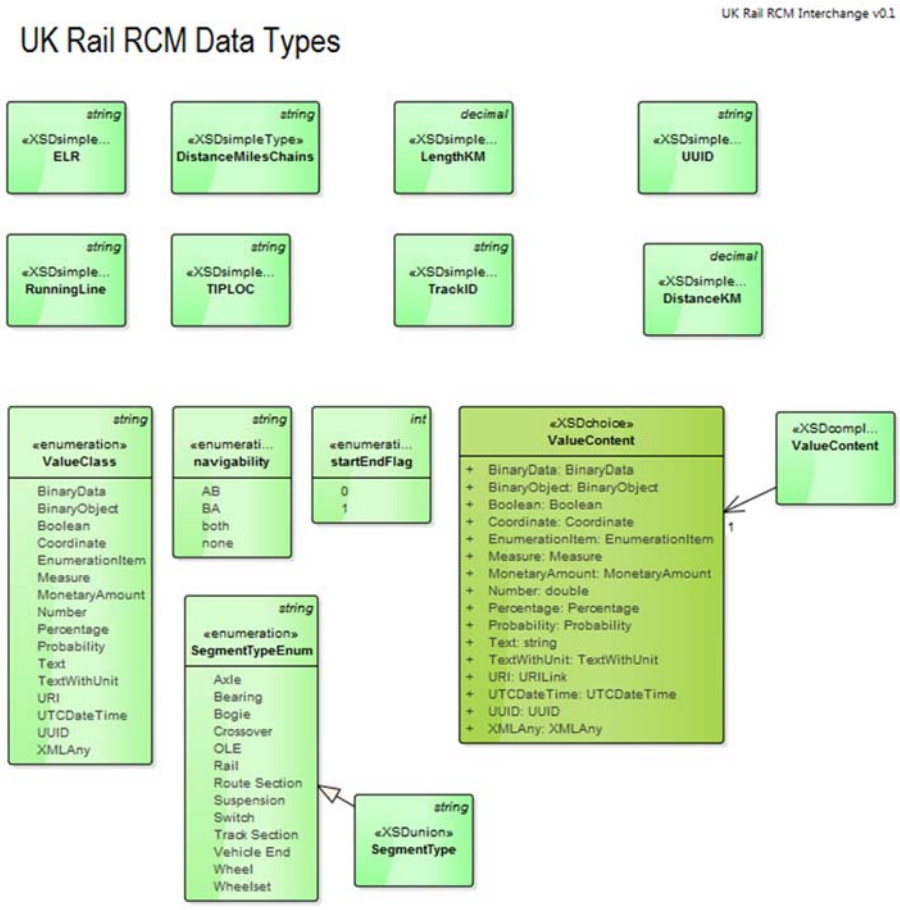
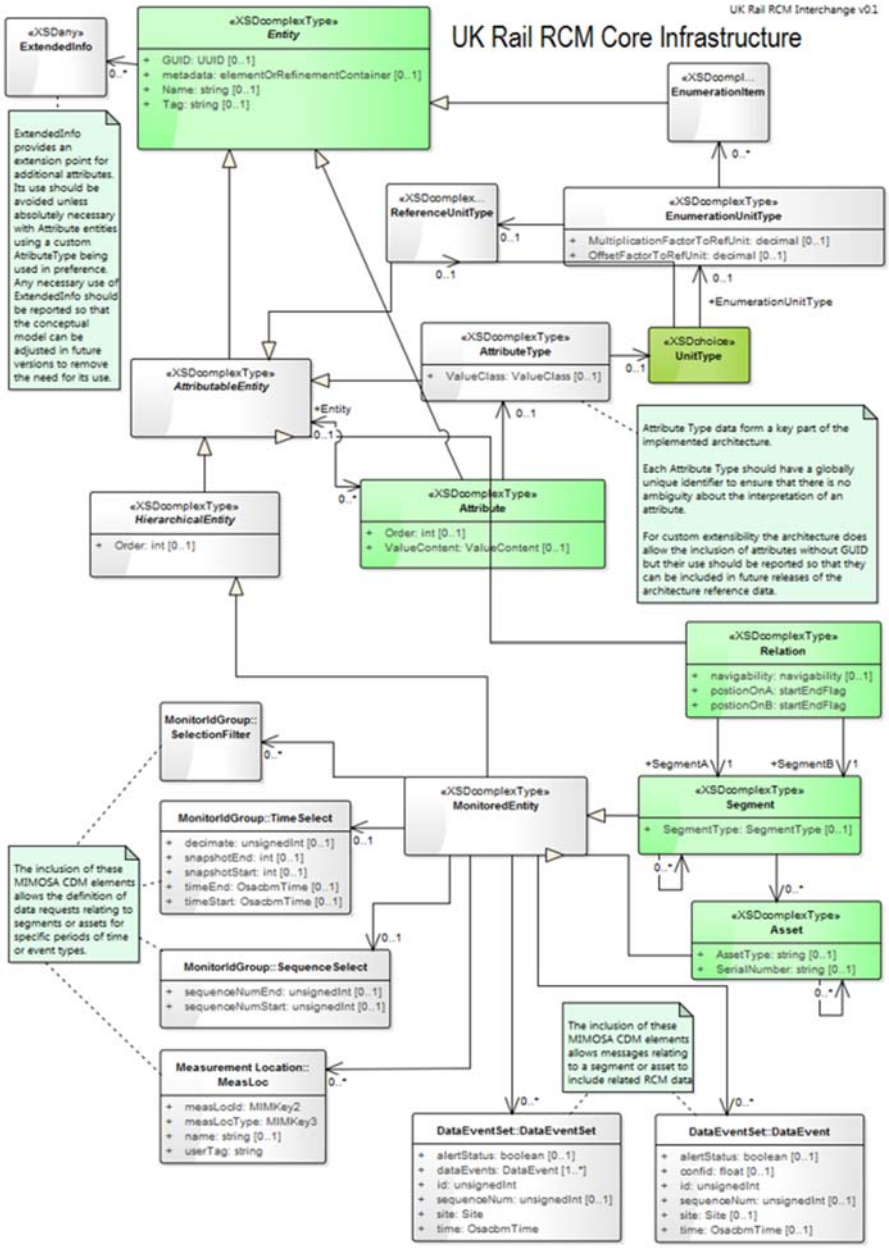


Figure 22 - RCM data conceptual data model



The data entities in these models are intended to be applicable to RCM data transfer for all types of rail assets. The data structure supports any kind of

hierarchy of assets in a single transfer and is extensible to include any type of data attribute that may need to be associated with the assets.

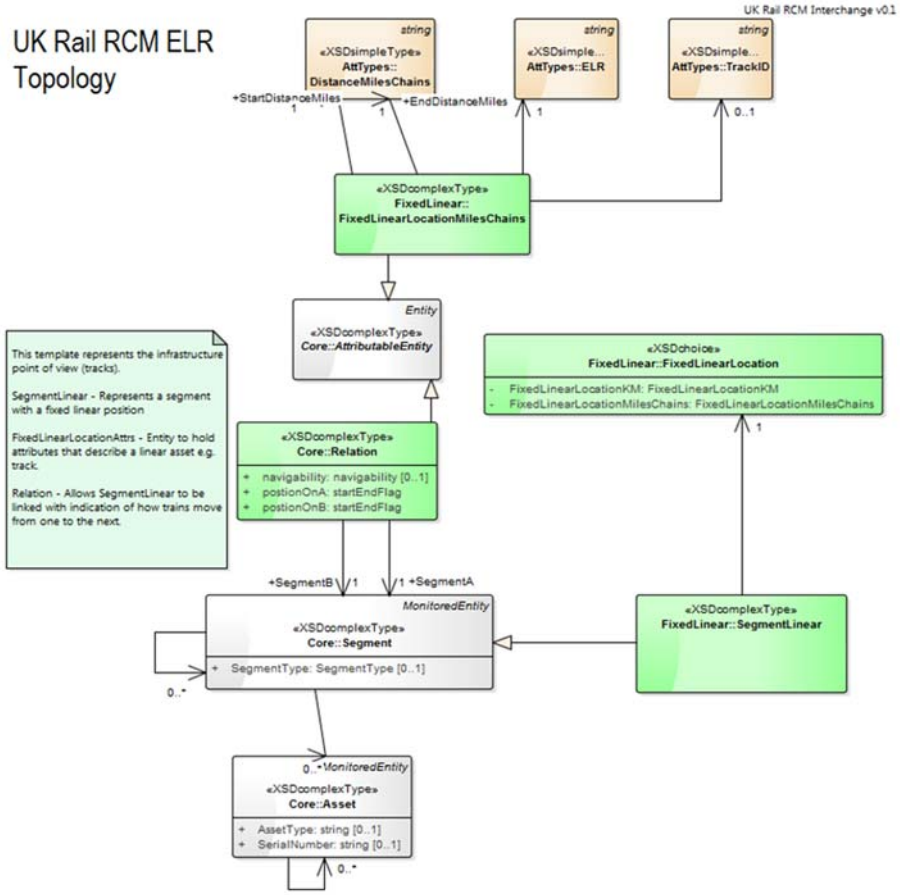
From these basic models, domain-specific ones can be generated by specialising. These specific models are known as Templates in ISO 15926. Sample specialised data models for particular asset types are shown in the following sections.

Network topology data models

The data models in this section are derivations of the generic data model for different views of the railway network. As well as being compatible with that data model, they are also compatible with the RailTopoModel generic view of railway networks as described in the *Networks and topologies* section.

Engineering topology template data model

Figure 23 - Network topology data model - engineering

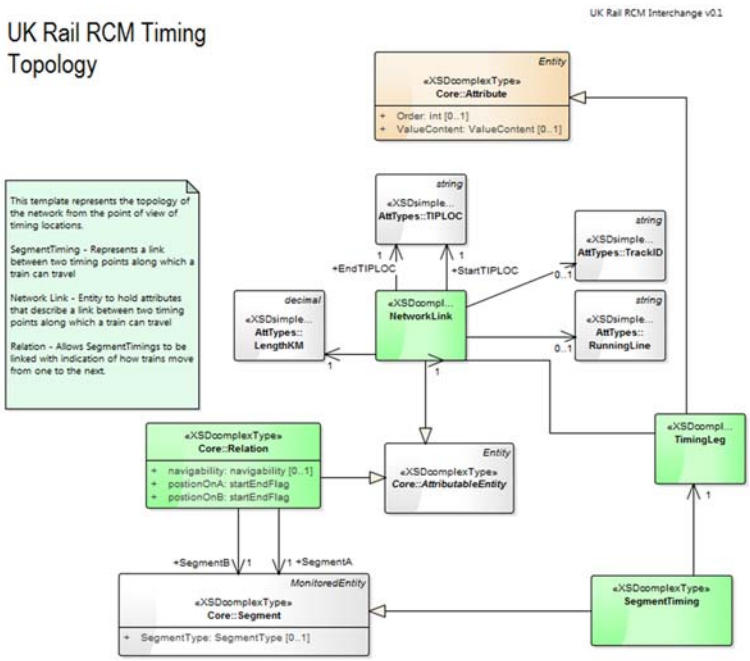


This template is for a network topology at the engineering level – dealing in track sections between switches & crossings and buffer stops, at the individual track level. It uses the ELR / TrackID / Miles:Chains method of linear referencing. It also includes a way of showing routing possibilities between track sections at junctions based on the configuration of switches and crossings.

This topology level corresponds to the 'micro' network view recognised by RailTopoModel.

Timing topology template data model

Figure 24 - Network topology data model – timing

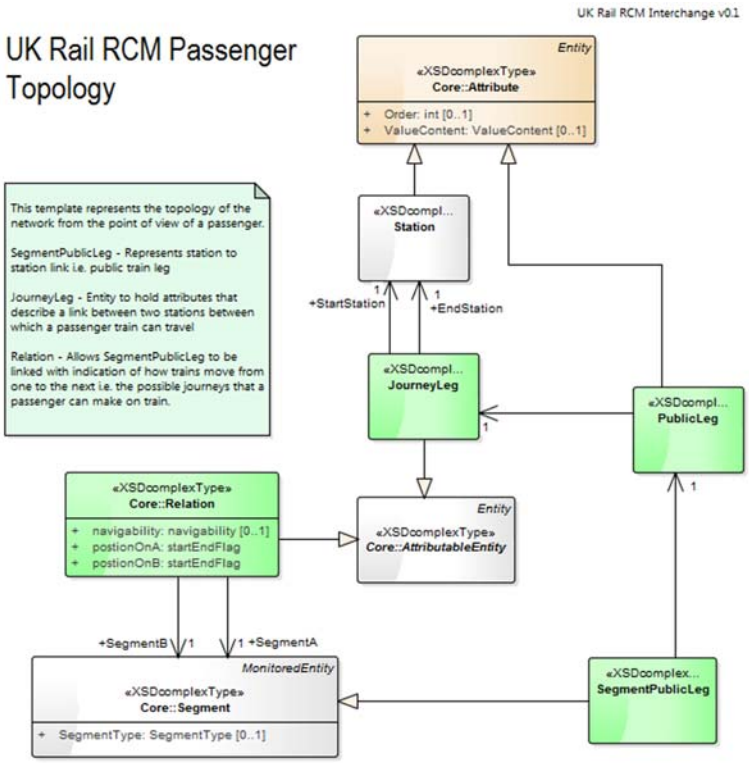


This template is for a network topology at the level used for timing and routing trains for the Working Timetable. Locations are timing points; links between locations are Network Links corresponding to timing running lines. The means exists to define allowable routings between network links based on junction configurations.

This topology level corresponds to the 'meso' level defined in RailTopoModel.

Passenger journey topology template data model

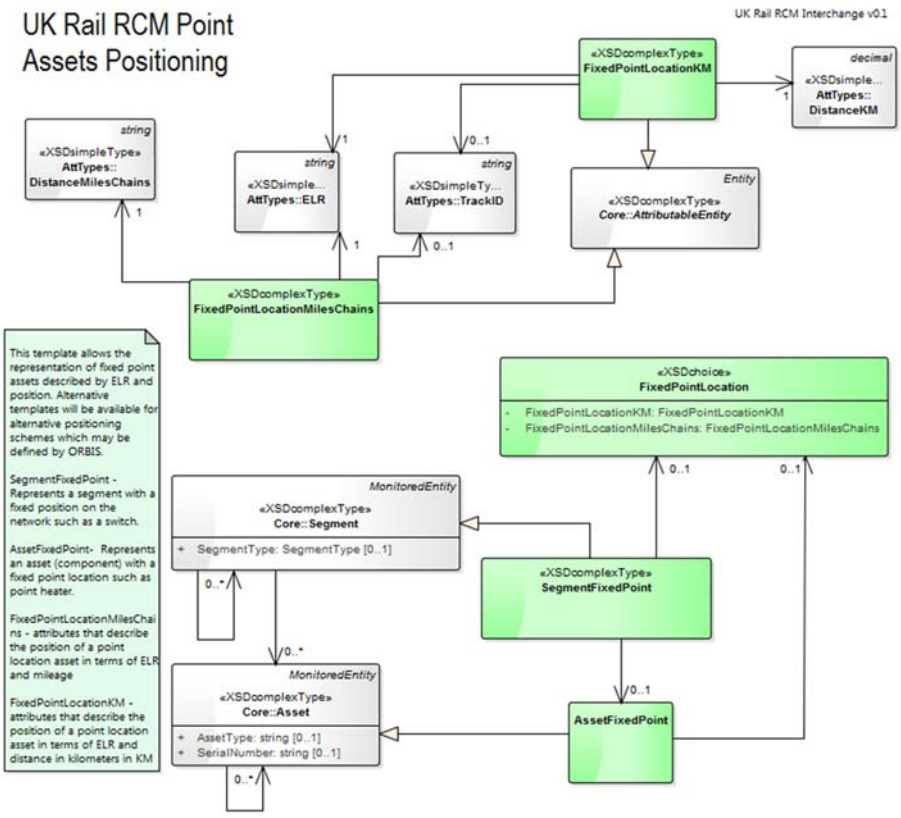
Figure 25 - Network topology data model - passenger journey



This template is for a network topology at the level used to plan and describe passenger journeys or the public timetable. Locations are Stations, usually identified by NALCO codes or CRS codes. The route links between stations are here called Public Legs. Routing possibilities can be shown.

Railway infrastructure template data model – point assets

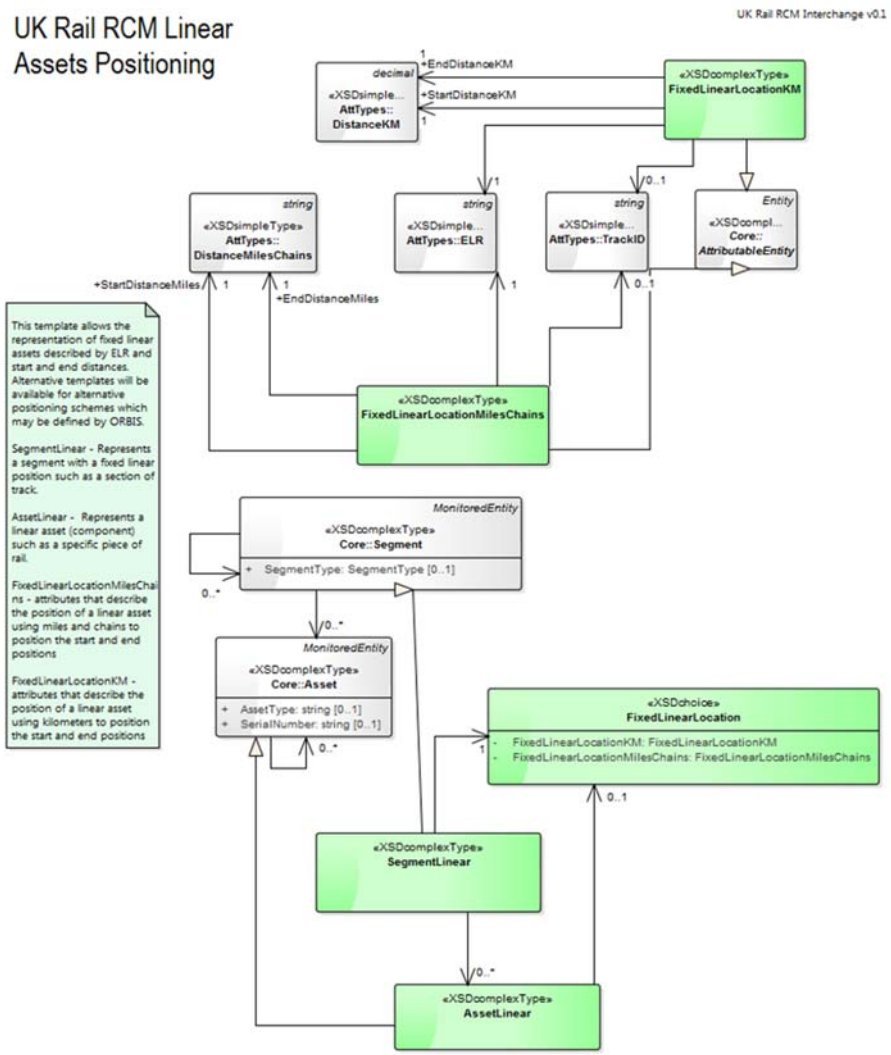
Figure 26 - Fixed point assets data model



This template data model is for fixed point railway assets, positioned on the network by ELR / miles:chains and possibly a track ID. It enables each such asset to be shown as an arbitrarily-complex nested hierarchy of components; and allows specific serial-numbered components to be associated with each part of the hierarchy.

Railway infrastructure template data model – linear assets

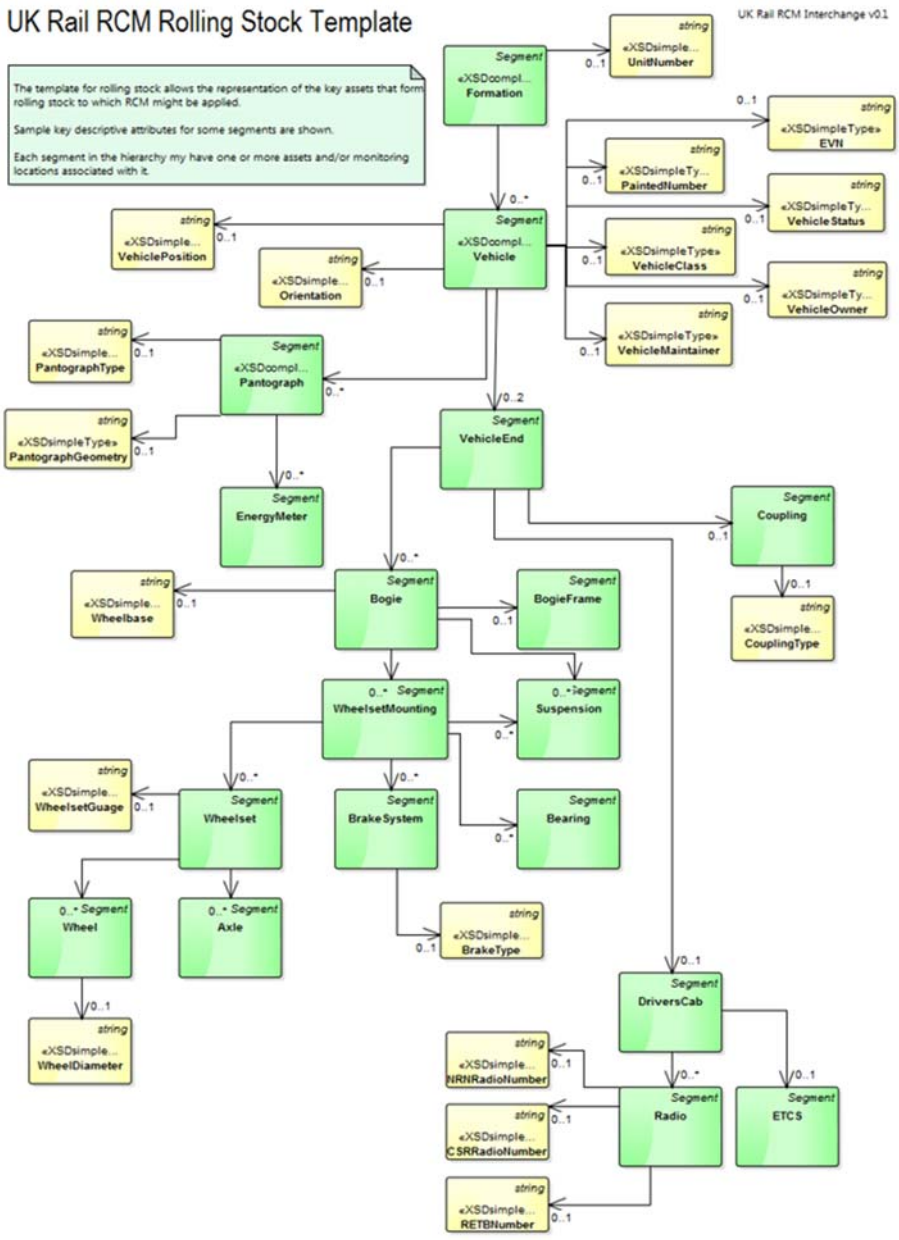
Figure 27 - Fixed linear assets data model



This template is for fixed linear assets, such as sections of track or OLE, identified by start and end track positions expressed as ELR / Miles:chains with an optional track ID. The asset itself can have a hierarchical structure of arbitrary complexity and can be described by any type of attribute. It can have serial-numbered components associated with any part of the hierarchy.

Rolling stock template data model

Figure 28 - Rolling stock template data model



This template data model is used for rolling stock. It supports vehicles, the breakdown of vehicles into the standard components recognised by RGS GM/RT2453 and RIS-2706-RST, and the amalgamation of vehicles into quasi-permanent formations such as multiple units. It should be noted that this is an example template for rolling stock: many others are possible with different levels of detail and aggregation appropriate to different uses and different types of rolling stock.

0099 The data architecture must support the sharing of conceptual data interchange data models and template data models.

0100 The data architecture should facilitate the creation of new templates for specific rail domains and data interchange requirements, compliant with the conceptual data interchange data models and existing templates.

Data interchange formats – datagrams

The data models described in the previous section indicate how data should be structured for interchange. In this section, we consider how these structures are implemented for actual data transfer.

This is done by means of datagrams, defined in the form of XML schemas [24] because this offers enough rigour and detail to be able to define all the elements of the datagram clearly and unambiguously, based on a shared data model. The schemas are mapped directly from the data models described above.

Whilst datagram formats are defined in this way, this does not mean that the data themselves need to be interchanged using XML, though there are advantages to doing so which are discussed in Section Data Interchange Methods. Any standard data interchange format is permissible provided that the overall structure of the data is compatible with the datagram format and the data items themselves are formatted as the data architecture requires.

The number of types of legal datagrams the data architecture needs to be able to support is very large, since it is the intersection of all the asset types, RCM data types, event types and ISO 13374 processing layers. The data architecture thus specifies Datagram Templates – standard layouts for data packets which apply to many different actual datagram layouts.

A datagram represents a standard packet of RCM data, together with the additional data items necessary to identify which asset or sensor it relates to, to describe what type of data it is and to add various metadata items which give additional information about the data.

An RCM Datagram contains the following elements, which are described in more detail in the *Datagram structure* section.

- Version and Schema identification. The way in which versions can be shown in datagrams is described in the section Header data – namespaces and versions; the Versioning and Releasing section recommends a versioning strategy for the architecture
- Asset or Sensor information. This indicates which railway asset or sensor the RCM data relates to. The Asset Data section gives details
- RCM data. This is the actual data RCM data payload being carried. It can be expressed in any of the data forms introduced in the section RCM data types. Data can be included within the datagram itself, or referred to from another source by a link
- Metadata. Each element of the datagram can be accompanied by metadata which provides context to it. The section Header data – namespaces and versions describes how the metadata are included in the datagrams; The Metadata section describes the types of metadata the architecture should support
- Extension points. The architecture defines formats and methods for those aspects of RCM data interchange that need to be standardised to realise cross-industry benefit. Other data specific to each interchange can be included as required by putting it in at these points. The section Header data – namespaces and versions shows where these appear and how they are to be used

Datagram structure

The basic structure of a datagram is shown in Figure 29.

A datagram consists of a *header* and a *body*.

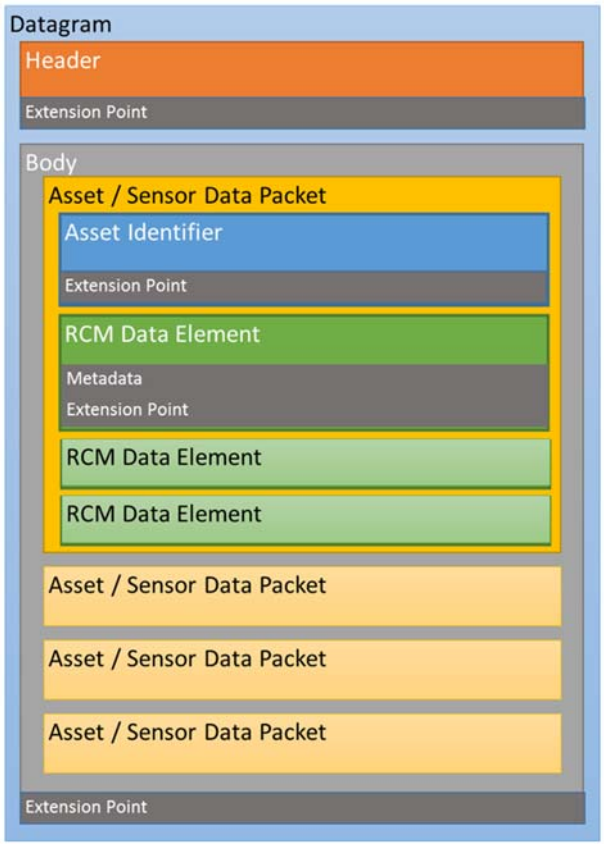
The header contains details of the version of the architecture and data format in use, a source indication and a timestamp (mandatory), plus any other (optional) metadata about attributes such as ownership, rights, and provenance; as described in the Metadata section. It also has an *extension point* which enables additional optional data items to be added as required by the providers and users of the data. These are not validated or restricted by the data architecture.

The *body* is optional, but if present will contain one or more data packets relating to an IT system, railway asset or a sensor (depending on the RCM data

and the level of the ISO 13374 stack concerned). It also has an extension point allowing interchange-specific data to be added.

Each data packet comprises a system, asset, or sensor identifier and one or more RCM data elements relating to the system, asset, or sensor.

Figure 29 - Datagram structure



Header data – namespaces and versions

Every datagram should show the context and version of the data it contains. This will enable users of the architecture to know the expected format of the data and to detect changes they need to be aware of.

0101 Several standard XML namespaces are involved in the definition of the architecture, depending on the exact data items included. The namespaces listed must include those used.

0102 Each element of the architecture must have a version number.

0103 The datagram header must include the architecture element version it conforms to.

Asset data

RCM data is gathered by sensors and relates to assets of some kind. The definition of the sensor or the asset are essential context for the data.

0104 Sensors and assets must be identified by a unique identifier as specified by the asset owner.

0105 The data architecture should enable Asset IDs to be translated between common representations by means of a lookup service. [3]

The system, asset, or sensor identifier defines which system, sensor or asset the data relates to. It will identify the asset or sensor using one of the methods described in the Railway Data Types – Infrastructure section, and may optionally also have other data items as required by the MIMOSA framework for a measurement on that asset or sensor type. It has an extension point which allows the parties to the data transfer to add any other data items they need.

The RCM data element contains data items as required by the RCM data type, which will be one of those described in the section RCM Data Types. This will contain the actual sensor or asset data. Optional Metadata items can include additional useful information relating to, for example, the quality, usage, provenance, dependability of the data, as described in the Metadata section. There is also an extension point which enables additional data items to be included as required by parties to the data transfer.

0106 The data architecture must permit non-validated data to be included at nominated extension points in datagrams.

0107 Data architecture managers should assess whether non-validated data items should be included in the data architecture model templates if there is a cross-industry benefit to so doing.

Mandatory and optional elements of a datagram

Although a datagram has placeholders for all of the components listed above, very few of them need to be used in any given interchange. This is to provide maximum flexibility as well as compatibility as far as possible with existing

3 The Harmonization – Running Behaviour and Noise on Measurement Sites (HRMS) Project [31] suggest some naming conventions for wayside train monitoring sites.

data interchanges, allowing them to be developed gradually to full compliance with the data architecture.

The only mandatory element of the datagram is the Header, which must have a timestamp and a source identification. This may be encoded in a filename if required.

The body of the datagram is optional: it is perfectly valid that a datagram may contain no data – such as to report no events. Once a datagram body is included, the asset / sensor identifier is mandatory but data content is optional.

These rules mean that data can be interchanged in familiar means such as .csv files or industry standard formats with little or no modification to the file naming convention or content format, and still be compliant in a basic sense with the data architecture.

0108 The data architecture must enforce the mandatory elements of datagrams.

0109 The data architecture must permit all optional elements of datagrams.

Example XML schemas

A set of XML schemas has been generated from the Data Interchange data models described in the *Data interchange data models* section. Each is a schema against which the relevant element of an XML datagram can be validated. The full set of XML schemas is shown in Appendix 1.

Some sample datagrams constructed according to these schemas are included in Appendix 2. They show what RCM data encoded in this way might look like. The sample datagrams have been shown in two ways – as XML conforming fully to the schema, and as CSV-format text files showing a subset of the data. The latter format also fully conforms.

Example schema – datagram

This example schema is for an RCM datagram, as described in the section *Data Interchange Formats – Datagrams*. In it can be seen the header, body and all the possible asset identification and RCM data elements as well as the extension points where additional user-specific data can be added.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="uk.rail.rcminterchange/0.1/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="uk.rail.rcminterchange/0.1/"
```

```
xmlns:ukrrcm="uk.rail.rcminterchange/0.1/" xmlns:osacbm="http://
www.mimosa.org/OSACBMV3.3.1" xmlns:dcterms="http://purl.org/dc/
terms/" elementFormDefault="qualified">

       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="StateDetection.xsd"/>
       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="PrognosticsAssessment.xsd"/>
       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="MonitorIdGroup.xsd"/>
       <xs:import namespace="http://purl.org/dc/terms/" sche-
maLocation="dcterms.xsd"/>
       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="HealthAssessment.xsd"/>
       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="DataManipulation.xsd"/>
       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="DataAcquisition.xsd"/>
       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="Configuration.xsd"/>
       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="ConfigRequest.xsd"/>
       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="AdvisoryGeneration.xsd"/>
       <xs:include schemaLocation="ukrrcmcore.xsd"/>
       <xs:element name="body" type="ukrrcm:body"/>
       <xs:complexType name="body">
              <xs:sequence minOccurs="0" maxOccurs="unbounded">
                     <xs:element name="AGDataEvent"
type="osacbm:AGDataEvent" minOccurs="0" maxOccurs="unbounded"/>
                     <xs:element name="AGPort"
type="osacbm:AGPort" minOccurs="0" maxOccurs="unbounded"/>
                     <xs:element name="ConfigRequest"
type="osacbm:ConfigRequest" minOccurs="0" maxOccurs="unbounded"/
>
                     <xs:element name="Configuration"
type="osacbm:Configuration" minOccurs="0" maxOccurs="unbounded"/
>
                     <xs:element name="DADataEvent"
type="osacbm:DADataEvent" minOccurs="0" maxOccurs="unbounded"/>
                     <xs:element name="DAPort"
type="osacbm:DAPort" minOccurs="0" maxOccurs="unbounded"/>
                     <xs:element name="DMDataEvent"
type="osacbm:DMDataEvent" minOccurs="0" maxOccurs="unbounded"/>
                     <xs:element name="DMPort"
type="osacbm:DMPort" minOccurs="0" maxOccurs="unbounded"/>
                     <xs:element name="Entity"
type="ukrrcm:Entity" minOccurs="0" maxOccurs="unbounded">
                            <xs:annotation>
                                   <xs:documentation>Entities
representing segments and assets</xs:documentation>
                            </xs:annotation>
                     </xs:element>
                     <xs:element name="HADataEvent"
type="osacbm:HADataEvent" minOccurs="0" maxOccurs="unbounded"/>
                     <xs:element name="HAPort"
```

```
type="osacbm:HAPort" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="metadata"
type="dcterms:elementOrRefinementContainer" minOccurs="0" maxOc-
curs="1">
                                <xs:annotation>
                                        <xs:documentation>Container
for Qualified Dublin Core Type metadata elements</xs:documenta-
tion>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="MonitorIdGroup"
type="osacbm:MonitorIdGroup" minOccurs="0" maxOc-
curs="unbounded"/>
                        <xs:element name="MonitorIdGroupList"
type="osacbm:MonitorIdGroupList" minOccurs="0" maxOc-
curs="unbounded"/>
                        <xs:element name="PADataEvent"
type="osacbm:PADataEvent" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="PAPort"
type="osacbm:PAPort" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="SDDataEvent"
type="osacbm:SDDataEvent" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="SDPort" type="xs:string"
minOccurs="0" maxOccurs="unbounded"/>
                        <xs:any namespace="##other" minOccurs="0"
maxOccurs="unbounded">
                                <xs:annotation>
                                        <xs:documentation>Provide an
extension point for custom application specific elements</
xs:documentation>
                                </xs:annotation>
                        </xs:any>
                </xs:sequence>
        </xs:complexType>
        <xs:element name="header" type="ukrrcm:header"/>
        <xs:complexType name="header">
                <xs:annotation>
                        <xs:documentation>Header for a UK Rail RCM
datagram. Required once for each message.</xs:documentation>
                </xs:annotation>
                <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:element ref="dcterms:created" minOc-
curs="1" maxOccurs="1">
                                <xs:annotation>
                                        <xs:documentation>Timestamp of
the message in ISO 8601 format</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element ref="dcterms:format" minOc-
curs="0" maxOccurs="1">
                                <xs:annotation>
                                        <xs:documentation>Format of the
data contained within the message</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element ref="dcterms:license" minOc-
```

```
curs="0" maxOccurs="1"/>
                        <xs:element ref="dcterms:rights" minOc-
curs="0" maxOccurs="1"/>
                        <xs:element ref="dcterms:rightsHolder" min-
Occurs="0" maxOccurs="1"/>
                        <xs:element ref="dcterms:source" minOc-
curs="1" maxOccurs="1">
                              <xs:annotation>
                                    <xs:documentation>Source of the
data.</xs:documentation>
                              </xs:annotation>
                        </xs:element>
                        <xs:element name="version" type="xs:string"
minOccurs="0" maxOccurs="1">
                              <xs:annotation>
                                    <xs:documentation>Version of
the architecture to which the message is compliant</xs:documen-
tation>
                              </xs:annotation>
                        </xs:element>
                        <xs:any namespace="##other" minOccurs="0"
maxOccurs="unbounded">
                              <xs:annotation>
                                    <xs:documentation>Provide an
extension point for custom application specific elements</
xs:documentation>
                              </xs:annotation>
                        </xs:any>
                  </xs:sequence>
            </xs:complexType>
            <xs:element name="rcmMessage" type="ukrrcm:rcmMessage"/>
            <xs:complexType name="rcmMessage">
                  <xs:sequence>
                        <xs:element name="header"
type="ukrrcm:header" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="body" type="ukrrcm:body"
minOccurs="0" maxOccurs="1"/>
                  </xs:sequence>
            </xs:complexType>
</xs:schema>
```

Example schema – rolling stock

This schema expresses the structure of a multiple unit as a Formation, made up of vehicles, made up of various key components. It is built from the rolling stock template data model described in the section *Rolling Stock Template Data Model*. The components and their data items conform to RGS GM/RT 2453 and RIS-2706-RST.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="uk.rail.rcminterchange/0.1/" xmlns:xs="http:/
/www.w3.org/2001/XMLSchema" xmlns="uk.rail.rcminterchange/0.1/"
xmlns:ukrrcm="uk.rail.rcminterchange/0.1/" elementFormDefault="quali-
fied">
	<xs:include schemaLocation="ukrrcmcore.xsd"/>
	<xs:element name="Formation" type="ukrrcm:Formation"/>
	<xs:complexType name="Formation">
		<xs:complexContent>
			<xs:extension base="ukrrcm:Segment">
				<xs:sequence>
					<xs:element name="Vehicle"
type="ukrrcm:Vehicle" minOccurs="0" maxOccurs="unbounded"/>
					<xs:element name="UnitNumber"
type="ukrrcm:UnitNumber" minOccurs="0" maxOccurs="1"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:element name="Vehicle" type="ukrrcm:Vehicle"/>
	<xs:complexType name="Vehicle">
		<xs:complexContent>
			<xs:extension base="ukrrcm:Segment">
				<xs:sequence>
					<xs:element name="EVN" type="ukrrcm:EVN"
minOccurs="0" maxOccurs="1"/>
					<xs:element name="PaintedNumber"
type="ukrrcm:PaintedNumber" minOccurs="0" maxOccurs="1"/>
					<xs:element name="VehicleStatus"
type="ukrrcm:VehicleStatus" minOccurs="0" maxOccurs="1"/>
					<xs:element name="VehicleClass"
type="ukrrcm:VehicleClass" minOccurs="0" maxOccurs="1"/>
					<xs:element name="VehicleOwner"
type="ukrrcm:VehicleOwner" minOccurs="0" maxOccurs="1"/>
					<xs:element name="VehicleMaintainer"
type="ukrrcm:VehicleMaintainer" minOccurs="0" maxOccurs="1"/>
					<xs:element name="VehicleEnd"
type="ukrrcm:VehicleEnd" minOccurs="0" maxOccurs="2"/>
					<xs:element name="VehiclePosition"
type="ukrrcm:VehiclePosition" minOccurs="0" maxOccurs="1"/>
					<xs:element name="Pantograph"
type="ukrrcm:Pantograph" minOccurs="0" maxOccurs="unbounded"/>
					<xs:element name="Orientation"
type="ukrrcm:Orientation" minOccurs="0" maxOccurs="1"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:element name="Pantograph" type="ukrrcm:Pantograph"/>
	<xs:complexType name="Pantograph">
		<xs:complexContent>
			<xs:extension base="ukrrcm:Segment">
				<xs:sequence>
					<xs:element name="EnergyMeter"
type="ukrrcm:EnergyMeter" minOccurs="0" maxOccurs="unbounded"/>
					<xs:element name="PantographGeometry"
```

```
type="ukrrcm:PantographGeometry" minOccurs="0" maxOccurs="1"/>
                                    <xs:element name="PantographType"
type="ukrrcm:PantographType" minOccurs="0" maxOccurs="1"/>
                              </xs:sequence>
                        </xs:extension>
                  </xs:complexContent>
            </xs:complexType>
            <xs:simpleType name="EVN">
                  <xs:list itemType="xs:string"/>
            </xs:simpleType>
            <xs:simpleType name="PaintedNumber">
                  <xs:list itemType="xs:string"/>
            </xs:simpleType>
            <xs:simpleType name="VehicleStatus">
                  <xs:list itemType="xs:string"/>
            </xs:simpleType>
            <xs:simpleType name="VehicleClass">
                  <xs:list itemType="xs:string"/>
            </xs:simpleType>
            <xs:simpleType name="VehicleOwner">
                  <xs:list itemType="xs:string"/>
            </xs:simpleType>
            <xs:simpleType name="UnitNumber">
                  <xs:list itemType="xs:string"/>
            </xs:simpleType>
            <xs:element name="VehicleEnd" type="ukrrcm:VehicleEnd"/>
            <xs:complexType name="VehicleEnd">
                  <xs:complexContent>
                        <xs:extension base="ukrrcm:Segment">
                              <xs:sequence>
                                    <xs:element name="DriversCab"
type="ukrrcm:DriversCab" minOccurs="0" maxOccurs="1"/>
                                    <xs:element name="Coupling"
type="ukrrcm:Coupling" minOccurs="0" maxOccurs="1"/>
                                    <xs:element name="Bogie"
type="ukrrcm:Bogie" minOccurs="0" maxOccurs="unbounded"/>
                              </xs:sequence>
                        </xs:extension>
                  </xs:complexContent>
            </xs:complexType>
            <xs:element name="DriversCab" type="ukrrcm:DriversCab"/>
            <xs:complexType name="DriversCab">
                  <xs:complexContent>
                        <xs:extension base="ukrrcm:Segment">
                              <xs:sequence>
                                    <xs:element name="Radio"
type="ukrrcm:Radio" minOccurs="0" maxOccurs="unbounded"/>
                                    <xs:element name="ETCS" type="ukr-
rcm:ETCS" minOccurs="0" maxOccurs="1"/>
                              </xs:sequence>
                        </xs:extension>
                  </xs:complexContent>
            </xs:complexType>
            <xs:element name="Radio" type="ukrrcm:Radio"/>
            <xs:complexType name="Radio">
                  <xs:complexContent>
```

```
                        <xs:extension base="ukrrcm:Segment">
                                <xs:sequence>
                                        <xs:element name="NRNRadioNumber"
type="ukrrcm:NRNRadioNumber" minOccurs="0" maxOccurs="1"/>
                                        <xs:element name="CSRRadioNumber"
type="ukrrcm:CSRRadioNumber" minOccurs="0" maxOccurs="1"/>
                                        <xs:element name="RETBNumber"
type="ukrrcm:RETBNumber" minOccurs="0" maxOccurs="1"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="Coupling" type="ukrrcm:Coupling"/>
        <xs:complexType name="Coupling">
                <xs:complexContent>
                        <xs:extension base="ukrrcm:Segment">
                                <xs:sequence>
                                        <xs:element name="CouplingType"
type="ukrrcm:CouplingType" minOccurs="0" maxOccurs="1"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="ETCS" type="ukrrcm:ETCS"/>
        <xs:complexType name="ETCS">
                <xs:complexContent>
                        <xs:extension base="ukrrcm:Segment">
                                <xs:sequence/>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:simpleType name="NRNRadioNumber">
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="CSRRadioNumber">
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="VehicleMaintainer">
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:element name="Bogie" type="ukrrcm:Bogie"/>
        <xs:complexType name="Bogie">
                <xs:complexContent>
                        <xs:extension base="ukrrcm:Segment">
                                <xs:sequence>
                                        <xs:element name="WheelsetMounting"
type="ukrrcm:WheelsetMounting" minOccurs="0" maxOccurs="unbounded"/>
                                        <xs:element name="Suspension"
type="ukrrcm:Suspension" minOccurs="0" maxOccurs="unbounded"/>
                                        <xs:element name="BogieFrame"
type="ukrrcm:BogieFrame" minOccurs="0" maxOccurs="1"/>
                                        <xs:element name="Wheelbase"
type="ukrrcm:Wheelbase" minOccurs="0" maxOccurs="1"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
```

```
	</xs:complexType>
	<xs:element name="Wheelset" type="ukrrcm:Wheelset"/>
	<xs:complexType name="Wheelset">
		<xs:complexContent>
			<xs:extension base="ukrrcm:Segment">
				<xs:sequence>
					<xs:element name="Axle"
type="ukrrcm:Axle" minOccurs="0" maxOccurs="unbounded"/>
					<xs:element name="Wheel"
type="ukrrcm:Wheel" minOccurs="0" maxOccurs="unbounded"/>
					<xs:element name="WheelsetGuage"
type="ukrrcm:WheelsetGuage" minOccurs="0" maxOccurs="1"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:element name="Axle" type="ukrrcm:Axle"/>
	<xs:complexType name="Axle">
		<xs:complexContent>
			<xs:extension base="ukrrcm:Segment">
				<xs:sequence/>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:element name="Wheel" type="ukrrcm:Wheel"/>
	<xs:complexType name="Wheel">
		<xs:complexContent>
			<xs:extension base="ukrrcm:Segment">
				<xs:sequence>
					<xs:element name="WheelDiameter"
type="ukrrcm:WheelDiameter" minOccurs="0" maxOccurs="1"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:element name="WheelsetMounting" type="ukrrcm:WheelsetMount-
ing"/>
	<xs:complexType name="WheelsetMounting">
		<xs:complexContent>
			<xs:extension base="ukrrcm:Segment">
				<xs:sequence>
					<xs:element name="Bearing"
type="ukrrcm:Bearing" minOccurs="0" maxOccurs="unbounded"/>
					<xs:element name="Wheelset"
type="ukrrcm:Wheelset" minOccurs="0" maxOccurs="unbounded"/>
					<xs:element name="Suspension"
type="ukrrcm:Suspension" minOccurs="0" maxOccurs="unbounded"/>
					<xs:element name="BrakeSystem"
type="ukrrcm:BrakeSystem" minOccurs="0" maxOccurs="unbounded"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:element name="Bearing" type="ukrrcm:Bearing"/>
	<xs:complexType name="Bearing">
		<xs:complexContent>
```

```
                        <xs:extension base="ukrrcm:Segment">
                                <xs:sequence/>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="Suspension" type="ukrrcm:Suspension"/>
        <xs:complexType name="Suspension">
                <xs:complexContent>
                        <xs:extension base="ukrrcm:Segment">
                                <xs:sequence/>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="EnergyMeter" type="ukrrcm:EnergyMeter"/>
        <xs:complexType name="EnergyMeter">
                <xs:complexContent>
                        <xs:extension base="ukrrcm:Segment">
                                <xs:sequence/>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="BogieFrame" type="ukrrcm:BogieFrame"/>
        <xs:complexType name="BogieFrame">
                <xs:complexContent>
                        <xs:extension base="ukrrcm:Segment">
                                <xs:sequence/>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:simpleType name="VehiclePosition">
                <xs:annotation>
                        <xs:documentation>Position of the vehicle within its
formation. Integer which increments one end of formation to other.</
xs:documentation>
                </xs:annotation>
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="Orientation">
                <xs:annotation>
                        <xs:documentation>Forward or Reverse. Forward if end
"1" is towards the front of the formation.</xs:documentation>
                </xs:annotation>
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="RETBNumber">
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:element name="BrakeSystem" type="ukrrcm:BrakeSystem"/>
        <xs:complexType name="BrakeSystem">
                <xs:complexContent>
                        <xs:extension base="ukrrcm:Segment">
                                <xs:sequence>
                                        <xs:element name="BrakeType"
type="ukrrcm:BrakeType" minOccurs="0" maxOccurs="1"/>
                                </xs:sequence>
                        </xs:extension>
```

```
            </xs:complexContent>
        </xs:complexType>
        <xs:simpleType name="BrakeType">
            <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="CouplingType">
            <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="Wheelbase">
            <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="PantographType">
            <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="PantographGeometry">
            <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="WheelsetGuage">
            <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="WheelDiameter">
            <xs:list itemType="xs:string"/>
        </xs:simpleType>
</xs:schema>
```

0110 The data architecture must make shared XML schemas available to all users.

Data interchange methods

The datagram formats described in the section *Data Interchange Formats – Datagrams* define the format and structure of data items to be interchanged using the data architecture – the 'what' of data interchange. In this section the 'how' is described: methods for requesting and transferring data.

Data providers and data users will benefit from adopting the standards for data format and asset identification defined in earlier sections, even if they use project-specific methods for requesting and providing data. However, larger gains to the broader industry will be realised by the use of the standardised approach to data interchange described here.

The methods described in this section support the Data Bus concept described in the section *Architectural Styles and the Data Integration Requirements* and following and conform to now well-established enterprise application integration principles for linking previously-separate IT systems.

Analysis of interchange methods

Since all interchange involves a medium of transmission, some of the methods will need an infrastructure to support them. Table 12 lists methods that will be compatible with the data architecture provided that the data being interchanged conform to the datagram requirements described above.

Table 12 - Data interchange methods

Data interchange method	Typical usage	Required infrastructure	Notes
File Transfer	Time-based or event-based data capture; summaries	None. Systems and users can use standard methods such as email to transfer files	Advantages: little investment required; standard file formats exist; source and destination do not need to be simultaneously available; indirect nature of access can enhance security. Disadvantages: require external processes to request, create and transmit the files; not suitable for many real-time applications.
Direct database access – centralised database	Standard or ad-hoc enquiries	Data access toolkit for the target database. Security access to the source database for each user. Database-specific n network access to each user.	Advantages: any type of query supported by the source database can be supported; performance is typically good. Disadvantages: needs data to be centralised; security risks to source data; end user knowledge of database structure required; tight coupling makes it harder to change database structure; each database needs to be dealt with individually; networking requirements can be onerous and hard to make reliable.
Web-mediated database access – centralised database	Standard or ad-hoc enquiries	Web application on top of source database. Security, user access control Web network access	Advantages: web application may offer extra functions such as data export. Disadvantages: centralised data; no standard for web applications; no standard for export formats; reports / analysis restricted to capability of web application.
Direct database access – distributed or federated database	Ad-hoc enquiries across several databases	Query-resolution framework: repository of data locations; RDF data store.	Advantages: as for direct database access; additionally, user does not need to know exactly where data are stored. Disadvantages: as for direct database access; standard SQL queries will not be possible.

Table 12 - Data interchange methods

Data interchange method	Typical usage	Required infrastructure	Notes
Web-style request and response	End-user querying tools or analytic applications	REST-style or SOAP-style web services interface with broker to route requests to appropriate sources	SOAP-based or REST-based approaches are both standard. This style allows web applications to query any connected RCM data source using a standard mechanism defined by the architecture. Advantages: standard web protocols allow access to all RCM data; all architectural features available. Disadvantages: requires standard naming scheme for assets, plus broker system.
Message request and response	Service-based applications, including real-time.	ESB-style messaging infrastructure supporting request or response, publish or subscribe, and asynchronous messaging styles.	The most capable method – encapsulates all the previous styles as well as offering real-time services which can be combined into arbitrarily sophisticated applications. Disadvantages: as for Web-style request, plus requires ESB infrastructure and user management.

MIMOSA request and response types

With the various request / response interchange techniques listed above, there are several different styles of interaction the architecture supports. These are based on the techniques supported by MIMOSA [14].

Figure 30 - Interaction styles

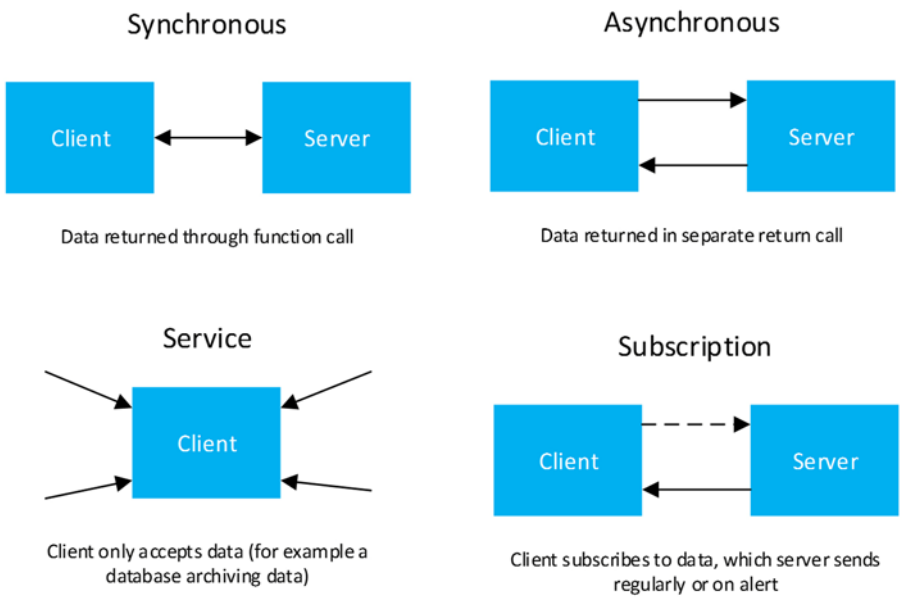


Figure 30 shows these interaction styles diagrammatically. Their application is described further in Table 13.

Table 13 - Interaction styles and applications

Style	Sample application	Notes
Synchronous	High-performance real-time data access.	Client-server access is of this type. Its main characteristic is that the client can do no further work until the server responds. This represents tight coupling and so should be avoided unless necessary.
Asynchronous	Less time-critical requests – e.g. overnight requests for data update.	Standard messaging protocol in which the client makes the request but does not wait for the server to respond. The client can carry on with other tasks; it is alerted when the server has the response ready.
Service	Database storage	Any external gateway such as email would also conform to this style.

Table 13 - Interaction styles and applications

Style	Sample application	Notes
Subscription	User-facing web application which monitors specific assets.	This is the 'publish / subscribe' style. The client registers its interest in particular data items with the server and is alerted whenever new data are ready.

The service-oriented approach

From the analysis in the previous two sections and the overall requirements for the data architecture gathered in earlier work, it is clear that the best match is offered by a service-oriented approach using a Data Bus. This supports the web-style and message request and response styles as well as all the MIMOSA interaction styles.

In the service-oriented approach, each connected data source IT system offers a number of software services to any other connected system. The services are invoked by means of a standard request from a connected system; and the results of the services communicated back via a response.

The software service may do any of these types of task, at a minimum. Any other type of service can be conceived and connected using the same mechanisms as these:

- Acquire and provide data from a sensor or logger
- Get data from an external source such as a weather forecast service
- Carry out an operation such as send an email, update a database or create an audit trail
- Process data included or referenced in the request and return the answer
- Store data in a repository or retrieve data from a repository.

The technologies of the World Wide Web have proven themselves very useful as a medium for the handling of requests and provision of responses, to the extent that 'Web Services' are now a standard mechanism for providing this type of service. Indeed, the World Wide Web itself is an excellent example of the possibilities opened up by the application of a simple request and response data protocol. The main reasons for this success are:

- The underlying protocol (hypertext transmission protocol, http) is an open standard, very simple and well-defined

- There is no dependence on any specific hardware or software platform
- It uses well-known and common networking technologies and is therefore reliable
- It is scalable to high data volumes and usage rates.

0111 The data architecture must support a service-oriented approach.

0112 The data architecture should use standard web technologies to provide services.

Requests and responses

Each interaction of any data user with the data architecture will comprise a series of requests to data providers and a response to each from each provider. The request will take the form of an http request to a specific URI which will define the service required and the parameters or input data required by the service; the response will be an http message containing result data, result status or error message and possibly some additional data items indicating other related services now available.

The request will typically include details of the IT system, asset, sensor or logger from which the data are required, the time band for which data should be provided, other restrictions such as data type, sensor type or location, and stipulations on how the results should be returned.

Result data may be returned in the body of the response or (more likely for large returned data sets) as a URI link to the location of the data. Elements of the response will indicate the format of the returned data.

Linked result data may be stored in a different network location than where the service itself is hosted. This may particularly be true for high-volume data such as streaming video or audio, which are likely to be made available on specialist servers.

In some cases where processing the request may take some time, the initial response will be an acceptance of the request, typically containing a link to the final results or to another status service. The requestor will query this link to see if the results are ready; if so, they will be returned; if not, a 'pending' response will be sent.

All requests and responses using the data architecture must use the data formats and concepts described in the Data Interchange data models to Data Interchange Formats – Datagrams sections. It will be the task of the data requestor to make sure that requests are formatted correctly; and of the data provider to do the same for the results.

0113 The data architecture must support a request / response style of operation.

0114 The data architecture must support data being embedded in responses to requests.

0115 The data architecture must support data being linked to by URI contained in responses to requests.

0116 The data architecture must support asynchronous responses and polling for request status. Asynchronous responses must include a URI link to the location of the results when complete.

0117 The data architecture must support streamed audio, video and other 'live' data.

Request and response style

There are several competing methods for implementing a service-based API using WWW technologies. They differ in the approach they take to defining the format and content of the requests and responses and have different characteristics which suit them to different types of application.

The two main approaches appropriate to the RCM data architecture are 'Message API' (of which an implementation based on the Simple Object Access Protocol (SOAP) would be typical) and 'Resource API' (sometimes, incorrectly, called 'REST-based'). The key differences between the two are outlined in Table 14.

Table 14 - Web service API approaches

Consideration	Note	Message API	Resource API
Basic operation	How does the service work?	A request for the service (name + parameters, including asset / sensor referred to) is encoded as an XML datagram and sent to a URI. The service returns another XML datagram which is unpacked by the user.	A request is sent to a URI which identifies the 'thing' about which data is requested (an asset or a sensor). HTTP methods such as GET or PUT are used to query or update data. The service returns a web response including the data or a reference to it.
Ease of Setup	How quickly / easily can a new service be added?	All the components of the API need to be defined fully before the service can be published and used.	The API can be built up gradually and can change during development without causing too much disruption
	How quickly and easily can a new data user be added?	Coding the call and decoding the response requires some programming effort. However, programming tools can often generate the code automatically based on the specification of the service.	Standard http queries and responses mean that data users can start simply and service is easy to understand. However, there is usually no simple automation of user-side code.
Security	How can transactions be made secure?	Standard web security protocols can be used. There are also advanced security standards such as WS-Security.	Standard web security protocols apply: https and web authentication.
Flexibility of Return Data	What formats can data be returned in?	Each service typically has one data return format, usually XML.	Data return types can be negotiated using standard http 'content negotiation' and can be very flexible.

Table 14 - Web service API approaches

Consideration	Note	Message API	Resource API
Error Management	What types of error can occur and how are they described?	Each service typically returns its own error messages, which may be difficult for end users to understand.	Standard http status messages such as 200 OK, 202 Accepted, 404 not found or 503 Service Unavailable.
Discovery of Features	How easy is it for a new user to find out how to use the service?	The basics of the service can be determined by looking at the service definition data; manual documentation is required for full understanding.	Well-designed services a) describe what features are available; b) conform to http standards. These support user-driven discovery of what the service does.
Degree of Coupling	How much does the data user need to know about the service?	Quite a lot. Messages must have the correct parameters or will fail with an error.	Relatively little. The URI encoding scheme needs to be known to find a resource; after that, repeated querying will enable the user to find out what the service does.
Work to Maintain	How difficult is it to keep up to date?	Fairly difficult. Any change to the service specification (such as a new parameter) will need all users to update their code in sync to match.	Fairly easy. Well-designed URI schemes are resilient to change, so data service can be updated without all users needing to update unless they need the new features.
Typical Implementation		SOAP-based XML	REST-based http

The Resource API approach generally suits the requirements of the data architecture better than the Message API approach, for these main reasons:

- Users of the architecture will be from many different organisations with different IT capabilities and development schedules. It is important not to force users to upgrade when changes occur to data services unless they need to.
- The same data may be used in different ways by different users using different code. It is therefore very useful for users to request the format they wish the data to be delivered in using a standard method such as http content negotiation.
- The ability for service users to discover the capabilities of the service by querying it supports the industry goal of opening the market up to new parties.
- In order to support a resource API, a standard method of addressing assets and requesting data is required based on a URI scheme. This scheme is discussed in Section Addressing and Referencing RCM Data and Assets.

0118 The data architecture should support a Resource API using http methods and responses

0119 The data architecture should support content negotiation to determine data return formats

0120 The data architecture should encourage user-driven exploration of service capabilities through self-documenting responses.

0121 Managers of the data architecture should minimise the degree of coupling between user and service code versions by careful version management.

Files

For the purpose of the RCM data architecture, a file is a named data set containing zero or more datagrams, structured in a published format based around the Datagram Structure described in Section Data Interchange Formats – Datagrams. Data can be exchanged in any recognised file format, whether as part of data requests or data responses. Recognised formats are:

- Existing industry standard file formats. The datagram structure has been defined in a way which should make these compliant with the data architecture

- New file formats which are compatible with the datagram structure and which can be given a recognised MIME type to identify their format. Typical such formats (in roughly descending order of desirability) are:
  - application/xml. This format gives maximum flexibility and clarity, though it is verbose and comparatively complicated to generate. The big advantage of XML is that it can be validated by the producer and the consumer to verify that it conforms to the published schema, thus ensuring that data are of the correct format.
  - application/json. This is a popular format for data to be consumed by end-user code such as web applications. It is more compact than XML but does not enable the same level of validation.
  - application/txt. This is a tab-separated text file format suitable for direct import into MS-Excel or other tabular processing packages.
  - application/csv. This is a comma-separated values file also suitable for direct import into MS-Excel. However, it is less reliable owing to variability in the ways that data items can be shown in this format.
- Standard MIME types for multimedia files such as audio or video. Recommended standards would be the open standards such as Ogg Vorbis; common but proprietary standards such as .mp4 would also be acceptable.

A file may be attached to the response or may be provided in the form of a URI link to a file stored separately: this is particularly likely to be the case for multimedia files or other large data files.

A key requirement of the data architecture is that it strongly encourages open and standard file formats and rejects proprietary, binary or otherwise opaque file formats.

0122 The data architecture must support the delivery of data via files.

0123 The data architecture must ensure file formats conform to its published standards.

0124 The data architecture must tolerate data being transferred in existing rail industry standard file formats (such as .CIF timetable format).

0125 The data architecture must ensure that files have an identified MIME type appropriate to its actual data type.

0126 The data architecture must ensure multimedia files are in open formats as far as possible.

Messaging

Figure 31 - Message-based interchanges

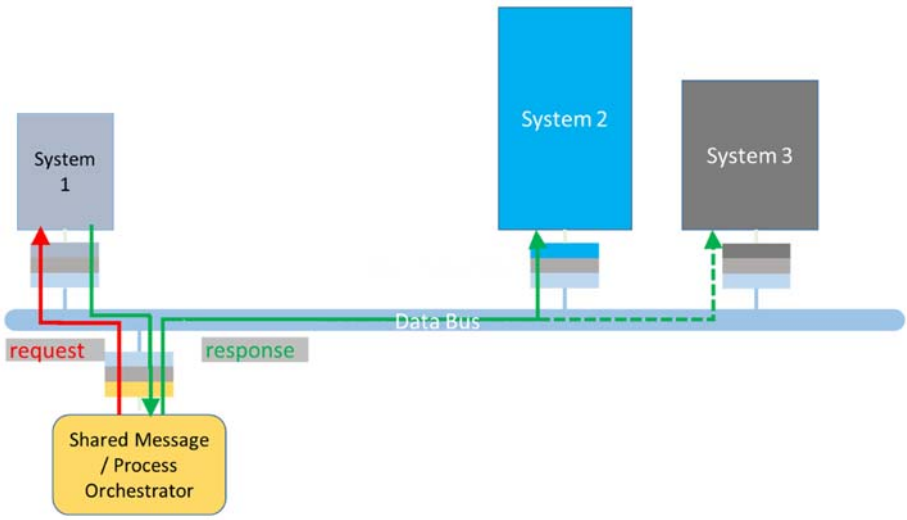


Figure 31 Message-based interchanges shows a typical usage of a message-based approach to acquiring and distributing data. It assumes the presence of an Enterprise Service Bus data management infrastructure to provide the routing facilities.

Once such infrastructure is in place, a number of useful messaging patterns become available to support distributed business processes. Although these are not strictly functions of the data architecture – they are more to do with an application architecture – it is worth considering them for their value in making business use of the service-oriented approach. These patterns are all described in Enterprise Integration Patterns [25] which is the reference document for this approach to integrating business processes. The most useful are listed in Table 15.

Table 15 - Useful messaging patterns

Pattern	Application / Purpose	Value for Cross Industry RCM
Request-Reply	Requesting information from a provider and getting a response. The request / response can be synchronous – i.e. the requestor waits for the response – or asynchronous – i.e. the requestor carries on with other work while the response is being prepared and deals with it when ready.	The basic mechanism for requesting and returning data
Guaranteed Delivery	Ensuring that messages (whether requests or responses) are received even if intermediate systems fail. This is done by storing the message and automatically re-trying it till successful	Transmission of alerts or alarms that have significant impact on safety or availability, so must get through.
Event Message	Informing receivers of the occurrence of an event	Live monitoring of assets.
Claim Check	Providing a link to data stored externally rather than including it in the message itself. This improves system performance.	Communicating links to, for example, video of a recorded incident.
Publish-Subscribe	Distributing messages to a list of recipients who have requested to be informed.	Broadcasting alerts and alarms to parties who have expressed the need to be told.
Recipient List	Sending a message to a list of recipients which is determined by the content of the message	Communicating alerts or alarms to different parties depending on their severity or criticality of the asset involved.

Table 15 - Useful messaging patterns

Pattern	Application / Purpose	Value for Cross Industry RCM
Process Manager	Managing multi-stage business processes by means of messaging.	Gathering data from multiple data sources in response to an alert or alarm from one source; deciding what to do based on the overall results of this data gathering.

0127 The data architecture should provide a messaging infrastructure to support message-based interactions.

0128 The data architecture should support standard Enterprise Integration architecture messaging patterns: request/reply, guaranteed delivery, event message, claim check, publish-subscribe, recipient list, process manager.

Addressing and referencing RCM data and assets

Referencing rail RCM entities – Cool URIs

The resource API approach described in Section Request and Response Style is based around a URI scheme by which 'resources' can be identified. A resource can potentially be any entity with meaning in the rail RCM context: a section of track, a signal cabinet, a rail vehicle, a data logger, a connected computer system.

While in theory any URI will suffice for an entity providing it is unique, good practice suggests the use of 'cool URIs' which are permanent for any permanent entity [26]. This means that they do not change because of changes to the underlying web server or database or because the resource they refer to has ceased to exist or been renamed, renumbered or moved.

Opaque and transparent URIs

It is possible for more than one URI to reference the same entity. This may be helpful in creating both 'opaque' and 'transparent' URIs for an entity. An opaque URI is one that identifies the resource but gives no clue to human readers about what type of resource it is or what other resources it might be

related to. A transparent URI is one which has a clear structure and hierarchy which encourages users to explore the hierarchy and find information about related and linked resources.

Since every entity in the rail RCM domain has a UUID, a straightforward way to generate an opaque URI for it is to use its UUID.

Hierarchical structure of transparent URIs

The URI scheme for each type of resource should follow a hierarchical scheme which reflects that of its domain in the 'real world'. For example, the top level subdomains of a resource hierarchy for rail resources might have values `/infrastructure`, `/rollingstock` and `/operations` (or abbreviated versions thereof). Other considerations for identifying top-level subdomains might be the owner of the data or the IT system holding the master reference, though these should not be specifically identified as they may change and therefore breach the 'cool URIs' rule.

Lower levels would indicate subdivisions, such as `/rollingstock/pax_unit_by_class/150/150_103` for a specific diesel multiple unit, or `/rollingstock/vehicle_by_evn/937034512342` for a specific locomotive.

Resource groups

In the transparent URI scheme, requesting a partial URI of a resource should return a list of the resources available, with their URIs. So, taking the example from the previous section, requesting `/rollingstock/pax_unit_by_class` would result in a list of classes of passenger units, with a URI for each one; and requesting `/rollingstock/pax_unit_by_class/150` would result in a list of all class 150 units.

Return formats

Http content negotiation should be used to control the format of the returned results. URIs should not contain a file extension or any other indicator of what format the resource will be returned in.

Use of http response codes

Http response codes can be used to provide useful information to requestors of information via a resource API. The full behaviour of the resource API will need to be fleshed out in design, but from an architectural point of view the following principles should be adopted for http response codes:

In case	Return code
Successful response	200 OK
Request acknowledged, processing – please wait for response	202, with URI of final response or request ID in the body
Item has moved	301 with URI of new location in the body
Item has been deleted	410 GONE
Item never existed (invalid request)	404 NOT FOUND

Querying data

Resource APIs can use http requests to query web services. This generally requires the addition of some query parameters such as date range / location range / data type to a resource identifier.

Standard http query syntax should be used for this purpose, with parameters attached as key=value pairs to the URI. The data architecture must define valid query parameters for each service.

There will be common query formats which are used frequently. It should be possible to use an abbreviated syntax for these, such as by using stored queries identified by a key=value pair such as 'query=latest_alerts'.

0129 The data architecture must support opaque URIs for all resources

0130 The data architecture should support a transparent URI scheme for all resources

0131 Data architecture managers should specify a transparent URI scheme that minimises the likelihood of change to resource URIs.

0132 Data architecture managers should specify responses to resource group URIs to identify resources in the group.

0133 The data architecture must support http content negotiation as the means of determining the return format of data requests.

0134 The data architecture should use http response codes to provide useful responses to users based on the state of data.

0135 The data architecture should support the use of http query strings to include query parameters.

0136 Data architecture managers must define a consistent scheme for query parameter naming for standard query components such as locations, dates, times, asset IDs, RCM data types.

0137 The data architecture should support the use of stored data queries to simplify the querying of frequently-used data.

Security and access control

Introduction

The data architecture is intended to connect together many railway IT systems which have significant security concerns of their own because of their important role in managing critical UK rail assets and infrastructure. It is thus important that the data architecture does not increase the exposure of these systems to unauthorised access of any kind.

The exact disposition and operation of the railway represent information that should only be made available to authorised viewers.

Contributors to the data architecture will have concerns on the security and commercial confidentiality of data they provide.

Important decisions relating to the availability and capability of the railway network and of the service it offers to passengers may depend on the RCM data transferred using the data architecture. Verifying the integrity and completeness of that data is therefore important.

The data architecture itself will become an important vector for rail data. It is important that it is itself robust and secure from external attack.

Securing existing railway systems and connected users

Each existing railway IT system and user system connected to the data architecture makes the connection via its data adapter (see the section Shared IT Elements). It is therefore the security of these adapters that determines the exposure to threat they receive from the data architecture.

The following design principles should be adopted to maximise this security:

- All interactions must use https with TLS security
- No other services should be open on the web server hosting the web service
- The web server should validate all URIs and query strings to protect against code injection and SQL injection attacks

- The web server should validate query parameters for sensible ranges and incoming URIs for payload size / content length to guard against denial-of-service attacks
- The web service should only be available to the rail industry network, not the public internet unless protected using a VPN
- Web services should only be available to authorised users, checked via a protocol such as 3-legged OAUTH

0138 The data architecture must use https with TLS security for all operations.

0139 Data adapter providers should limit the attack surface of data adapters connected to the data architecture to the services offered only.

0140 Data adapter providers should secure their web services against code injection and SQL injection attacks.

0141 Data adapter providers should verify web requests to guard against denial-of-service attacks.

0142 Data adapter providers should limit access to their web services to the rail industry network.

Authenticating users

Only approved users must be able to use the data architecture. The data architecture will need to keep a repository of users and which services they are allowed to use.

The data architecture should enable connected users – whether human or system – to use any of the services they are entitled to without needing repeatedly to sign in to each one.

0143 The data architecture must provide an authentication service to enable data adapters to verify the identity and authority of web service users.

0144 The data architecture must support the restriction of access of users to data they have been authorised to view or change.

Verifying data integrity

Datagrams transferred using the data architecture may be subject to tampering. To guard against this, they can be electronically signed. For XML data, the XML Signature method is available [12]; for data of other types, one of the metadata areas in the datagram may be used to contain a hash of the

data content using a hashing mechanism such as MD5 or SHA1 or, more securely, HMAC.

This type of hashing enables the receiver of the datagram to verify that a) the message has not been tampered with; b) it comes from whomever it says it does – i.e. it has not suffered a 'man-in-the-middle' attack.

0145 The data architecture must support the use of XML Signature or other content hashing method such as HMAC to verify the integrity of datagrams.

Encrypting sensitive data

Notwithstanding the use of secure web protocols and message signing, there may be a requirement in some cases for some or all of the content of a datagram to be encrypted in such a way that only the sender and receiver can decode it.

For XML datagrams, the standard XML Encryption method is available [13]. This enables specific elements of a datagram to be encrypted, and the encryption information to be sent with the message or separately.

0146 The data architecture must support the use of XML Encryption for specific datagram components.

Securing the data architecture

The data repositories and physical infrastructure that support the data architecture must themselves be kept secure to prevent destruction or tampering with the data and to maintain the availability of the services offered by the data architecture.

0147 Data architecture managers must ensure the physical safety of data architecture infrastructure and data.

0148 Data architecture managers must control access to data architecture data and services.

Audit trail

The data architecture must be able to keep an audit trail of all the data interchanges that take place using it and all attempted accesses to it from whatever source. This is to assist in the diagnosis of security and performance issues.

0149 The data architecture must support the logging of all access and usage.

Industry reference data requests

A particularly important set of requests and responses is those that will need to be provided by rail industry IT systems presently in existence or being developed, to provide data lookup services. These will be vital in ensuring that accurate and correctly-formatted data about assets, network topology and train services are available to all RCM data exchanges.

It is likely that over time an extensive catalogue of such services will be developed, particularly if the concept of providing reference data via this request / response method takes root, rather than the current methods which are typically based around periodic file dumps. However, our initial consideration of the use cases for the data architecture has resulted in a basic set of enquiries, summarised in Table 16.

Table 16 - Industry reference data requests

Data Enquiry	Note	Request	Response
Network Location	What is the track location corresponding to a given GPS position? Provided by a Network Model Service	GPS position. (Optional) direction of travel (Optional) timetabled running line (Optional) requested frame of reference	Railway location in requested frame of reference (say ELR / Miles / Chains) Track ID if known or calculable. Precision.
Train Routing	What was the actual routing of a given train? Provided by a train operations service	Train ID; Run Date Optional: List of known stopping locations Optional: Corresponding list of odometer readings	Full train timetable with planned and actual routings, planned and actual stop and pass times. Odometer offset and calibration correction.
Asset Composition – Drill-up	Which larger asset units are a given asset part of? Provided by an Asset Register Service or possibly a Network Topology Service	Asset ID and / or Asset UUID. Optional: date / time. For example the EVN of a rail vehicle.	Hierarchical list of the parents of this asset. For example, for a rail vehicle this would be the multiple unit or rake identifier for the vehicle's unit or rake, followed by the train consist of which this unit is a part at the given time.
Asset Composition – Drill-down	What is the component breakdown of a given asset? Provided by an Asset Register Service	Asset ID and / or Asset UUID. For example, the identifier of a set of points.	Hierarchical list of the child assets of this asset. For example, for a set of points, a list of the components of the set of points, with text identifiers and UUIDs for each one.

Table 16 - Industry reference data requests

Data Enquiry	Note	Request	Response
Vehicle Identification by EVN	What is the train consist of a train passing a trackside location at a given time, where some EVNs are known? Provided by a train identification service	Track Location Date + Time List of one or more EVNs – need not be complete.	Full consist of train, including all vehicles, with the EVN and the orientation of each.
Vehicle Identification by Network Location	What is the train consist of a train passing a given trackside location at a given time? No EVNs known Provided by a train identification service	Track Location Date + Time Optional: Track ID Optional: Direction of Travel	Full consist of train, including all vehicles, with the EVN and the orientation of each.
Train Service Identification by Network Location	What is the operational train passing a given track location at a given time? Provided by a train identification service	Track Location Date + Time Optional: Track ID Optional: Direction of Travel	Full train timetable with planned and actual routings, planned and actual stop and pass times.
Train Service Identification by EVN and Time	What is the operational train which includes a given EVN or list of EVNs at a given time? Provided by a train identification service	Date + Time List of EVNs – need not be complete	Full train timetable with planned and actual routings, planned and actual stop and pass times.

Table 16 - Industry reference data requests

Data Enquiry	Note	Request	Response
Stock Working	What are the future tasks to be undertaken by the identified rolling stock at a given time? Provided by a Stock Dispatching Service	Unit, Rake or Locomotive number; EVN Date + Time	List of trains to be operated and activities to be undertaken by the Unit / Rake / Locomotive (implied by the EVN) later the same day, with start and end locations and times; planned stabling location at end of day.
Crew Working	What are the future tasks to be undertaken by the train crew operating the identified rolling stock / operational train at a given time? Provided by a Crew Dispatching Service	Unit, Rake or Locomotive number; or EVN; or Train ID Date + Time	List of crew diagrams being worked on the train, and for each, the future workload to the end of the day.
Affected Trains	Which trains are affected by an issue at a given track location and time period? Provided by a train identification service	Track Location Start Date / Time End Date / Time (Optional) level of detail of response	List of trains passing or stopping at the specified location in the specified time window.

These services are not directly a requirement of the data architecture, but they are a vital contributor to its usability. The data architecture must be set up to facilitate their creation and operation. Some of the services may be available directly from connected reference systems, as they represent straightforward extracts or queries of existing reference data; others may need to be provided separately by new services using such reference data as input, but providing novel processing (as described in Section Shared Asset Relationship Services).

0150 The data architecture must support the creation of 'value-added' reference data services which offer complex mappings or interpretations of raw reference data.

Implementation

IT infrastructure requirements

Figure 5 shows the main component elements of the data architecture, with gradually increasing levels of data and process integration. Each of the integration levels implies a certain amount of shared infrastructure and data – i.e. IT artefacts that need to be managed by a third party aside from the producer and user of any given RCM data stream. In this section we outline what the IT infrastructure would need to be to support each of the integration levels.

Overview

Table 17 lists the data architecture levels of integration, showing for each one the requirement for IT infrastructure and possible ways in which that infrastructure might be provided – whether by enhancement or application of existing systems or by the provision of new shared IT systems.

Table 17 Infrastructure Elements for data architecture Layers

Table 17 -

Data architecture layer	Infrastructure requirement	Possible implementation
1 Standard Data Item Formats	Reference data store of asset types, data types, sensor types	Spreadsheet repository on a shared website (like sparkrail.org)
2 RCM data types	Unique ID generation and referencing system for rail assets and components. Reference data store of XML schemas and lookup data types (such as engineering units, and sensor types)	New shared UUID generator; possible use of existing UUID generator such as from Intelligent Infrastructure. Spreadsheet or database repository; web service.
3 Standard reference asset nomenclature	Industry reference data for moving and fixed assets for validation. Industry standard formats for representing assets / locations / trains	Lookups on industry reference systems (RSL / ORBIS / ITPS) or shared proxies that provide service. Rolling Stock RIS.
4 Datagram structures	Formal shared Conceptual Data Model and Realisations. Repository of valid XML schemas; online data validator; documentation on schemas.	Shared database model. Shared file repository. Open documentation (such as a wiki) for community of practice.
5 Interchange methods and transaction protocols	Enterprise Service Bus infrastructure. Security repository / industry-wide access control.	Re-use of existing ESB infrastructure such as NR ESB or LINX-TM; or new centrally-managed ESB.
6 Asset relationships	Shared ontology repository and access services. Enhanced reference data services provided by master system sponsors	New shared ontology server and web services. New sponsor-provided facades for industry systems; centrally-managed proxies for these systems.

More detail about these infrastructure elements is given in the following paragraphs.

Reference data store

To support the MIMOSA framework and the derived realisations of it for UK rail asset types, sensor types and data types, a set of reference data tables needs to be populated. These are relatively static tables, changing only when new types of asset template, sensor or data type are required by a project. They are also small tables. The whole data store can therefore be managed as a set of text files.

Industry reference data

The two concepts here – asset reference data and standard representations - are handled differently.

Asset reference data

Given the volume and rate of change of UK rail assets and components and the fact that certain industry systems are designated as the master data source for each type of asset, it is neither desirable nor realistic for there to be a shared data store for them all as part of the data architecture. We have therefore assumed that the system sponsors for the master reference systems (such as Rolling Stock Library (RSL) or track (ORBIS, ultimately)) will provide lookup functions for assets using services similar to those discussed in Section Industry Reference Data Requests.

Alternatively, there may be a case for Shared Proxies for some or all reference data, based on the concept discussed in Section Introduction. These would be centrally-managed servers providing easy-to-use standard reference lookup web services, based on periodic data extracts from the actual reference data sources.

Standard representations

Standard ways of representing UK rail notions such as track location, train ID, and vehicle structure, will be defined outside of the data architecture, by the sponsors of the systems or by an industry-wide specification document such as a Rail Industry Specification.

Shared conceptual data model and realisations

As the scope of the data architecture expands as projects use it, new realisations of the conceptual data model will be required to represent the new asset types, data types or interchange patterns required. To maintain syntactic compatibility of these realisations, they must all be based on the same conceptual data model and created using a standardised process of instantiating ('de-generalising') the model.

The conceptual data model itself therefore needs to be stored in a way that makes it suitable for this purpose. This might be using a UML model in a commonly-used format such as Enterprise Architect, though a more standards-compliant method might be to store it as an OWL ontology.

XML schema repository and namespace

The XML schemas are set up to realise aspects of the conceptual data model to support specific types of data exchange. They define what constitutes a legally-formatted datagram for that exchange. The great advantage of the XML schema language is that it allows any specific datagram to be validated automatically against the schema. It therefore acts as a 'gatekeeper' to support the maintenance of data quality for data transferred using the data architecture.

Each specific data interchange will likely be mediated by its own XML schema which will use and derive from the published architecture schemas. To enable this to be done in a robust way, the XML schemas must be available in a stable 'namespace': a URI which refers to the formal definition of the schema.

Associated with the formal XML Schema language will be textual comment which will be valuable for implementers of data interchanges.

For both these reasons it is important that these schemas are made easily available on robust and permanent web addresses for data integrators to use and to refer to in their code. This needs to be via a centralised website such as sparkrail.org or a website set up specifically for the purpose.

Enterprise service bus infrastructure

This infrastructure makes available the Data Bus and message-oriented approach to data interchange, with all the benefits described elsewhere in this document.

The GB rail industry already has several ESB implementations in place. Network Rail has its own ESB infrastructure, as well as a separate message-

based infrastructure for the LINX-TM programme. It would be preferable to re-use and extend one of these to host the data architecture's Data Bus elements, though a centralised separate ESB could be set up if this is not feasible. There are many possible ways of implementing an ESB, including open-source ones that can be deployed initially on a small scale and extended as required for relatively little cost.

Shared security and access control

Access to the Data Bus needs rigorous fine-grained access and security control. It is particularly important that the connection of centralised UK railway systems to the data bus via adapters does not compromise their security in any way.

This means the implementation of a shared security system in which any user of the data architecture can be validated and authenticated and an agreed level of access to the various connected systems managed.

There may be suitable authentication services already available in the industry which can be extended for this use.

Shared ontology

As the data architecture is developed and the facilities it offers move higher up the ISO 13374 hierarchy, the types of function being added will tend to require the merger of more disparate data types from different connected IT systems. The mapping of concepts rather than just data items becomes more important.

The best way to manage these mappings is the use of a shared rail ontology. This needs to be made available on a standard and stable URI (such as sparkrail.org) for the use of data integrators who will then be able to reference it in their own ontologies and data schemas.

Shared asset relationship services

Applications using the data architecture will come to require more complex and sophisticated information about the relationships between different elements of the rail network. Whilst we have anticipated that simple asset relationship queries will be met by the reference systems themselves (via data adapters and web services on the data bus) or by centralised proxies for these, it is likely that more complex enquiries will need specialist code to meet them.

To make the development and deployment of these queries faster and more reliable, it may make sense for them to be provided centrally on dedicated web

servers associated with the shared ontology and the reference data proxies rather than with the reference data systems themselves.

Other infrastructure elements

There are other elements of IT infrastructure which may justify central provision and management rather than being left to existing industry parties to provide. These may include:

- An open documentation / user community site. This would provide a repository for best practice, information sharing and support.
- Large-scale data stores, such as for bulk application logs, video and audio. These would support the storage of these 'big data' items and the linking to them from data architecture datagrams, so disconnecting the source systems from the demand for this type of data.
- Online data mapping tools and XML validation tools. These services would support developers in the creation of interfaces to the data bus and the testing of their adapters and message formats.

The analysis of the literature and development of the data architecture principles and requirements described in the preceding sections have generated some suggestions about the management activities that will need to be carried out centrally to develop and maintain the architecture. They relate particularly to the shared infrastructure elements and tasks described in Section Overview.

0151 The data architecture must offer a shared community documentation site.

0152 The data architecture must support the attachment of bulk data stores for 'big data' items such as logs, audio, video and large-scale RCM data.

0153 The data architecture should offer tool support to the developers of Data Adapters, including XML validation tools and mapping tools to assist in datagram schema generation.

Architecture repository management

A body will be necessary to manage the formal definition of the data architecture and to ensure that the artefacts which define it are readily available to users. This body will need to:

- Monitor the external standards from ISO, MIMOSA and others for changes which might impact on the data architecture

- Maintain the architecture Principles on which the data architecture is built to ensure that they keep in step with emerging IT best practice, technological opportunities and changes in the rail business environment
- Manage a Change Control process to gate-keep changes and extensions to the data architecture as it is developed, refined and extended through use in projects
- Manage any changes to UK rail industry standards and railway group standards based on the data architecture
- Manage the documentation and web assets which define the data architecture and make sure that they are available for implementers of the data architecture to use.

Architecture infrastructure management

Where shared IT infrastructure is used to support the implementation of the data architecture functions described in Section IT Infrastructure Requirements, a governance function is required to fund and provide for them and ensure that they are managed appropriately. This function will need to:

- Get industry support for the infrastructure investment required. This will mean soliciting and eliciting the needs of industry data sharers for the infrastructure, generating a business case, securing funding and setting up a development project.
- Work out the technical specification and non-functional requirements for the infrastructure (e.g. availability, performance, data quality indicators)
- Work out a contractual framework for its provision
- Procure the supply of the infrastructure
- Monitor the performance of the infrastructure
- Manage a Change Control process for enhancements and extensions to the infrastructure to meet new demands on it.

Certification and compliance

The duty to provide data services in accordance with the data architecture will be included in commercial agreements between providers and data users, based on the framework established in project T1010-02 [27]. Data providers in particular will need to be able to confirm that the services they offer are compliant with the data architecture.

This suggests a need for a Compliance body to manage the certification of compliance to the data architecture. This body will have the following functions:

- Setting up and maintaining a set of tests and tools by which data providers and data users verify their own compliance with the data architecture. These may be standard datagrams which should be able to be processed correctly; XML schemas which can be used to validate datagrams; dummy service endpoints that can be used to test interactions and performance.
- Maintaining a register of compliant data providers
- Adjudicating on disputes between data providers and users on the reasons for problems with their data interchange, where these are germane to the data architecture
- Verifying performance of data providers' services according to their published or agreed Service Level Agreements, in terms of availability, currency, performance, data quality etc.

Support for implementers

Wide adoption of the data architecture will depend on making it easy to use. New users will need to be supported in getting connected to the data architecture. This suggests a support organisation whose responsibilities will include:

- Managing worked examples of using the architecture and other user-oriented documentation
- Maintaining a user community website and wiki where best practice, users' experiences and support queries can be raised and answered; evangelising the use of this community documentation
- Assisting in users' efforts to define the form and scope of their data interchanges making best use of the data architecture's facilities
- Assisting in prototyping and proof-of-concept exercises by intending users
- Facilitating changes and extensions to the data architecture to meet emerging user requirement.

Versioning and releasing

The data architecture will be subject to change throughout its lifetime. Particularly where shared infrastructure or common data services are involved, changes to the data architecture could affect other users than those any change was made for, leading to enforced software upgrades, failures of data interchange and additional cost.

The elements of the data architecture must be version-managed with the goal of maximising both backwards compatibility (for example, where new data user code can use older data provider code) and forwards compatibility (for example, where a new version of a data service can still be used by data users using older client code), thus minimising the amount and rate of forced change for users.

There is a need for a body to regulate the versioning of the elements of the data architecture and to manage the release process in conjunction with the user community to minimise the disruption caused by change. The functions of this body will be:

- To assess all change requests for the data architecture to determine how they should be released and versioned based on the breadth of value, urgency of change and impact on the installed user base.
- To assist in defining the scope of changes so that they can be implemented with minimum disruption
- To manage a release schedule for new versions of the data architecture and to ensure that it is clearly communicated. This involves indicating what new functions are available to ensure that they are used; and indicating where breaking changes are likely to occur to existing connected systems.
- To manage a deprecation schedule for obsolete or superseded items which gives users ample time to migrate their code and data formats away from them.
- To liaise with the sponsors of reference data source systems to ensure that changes to these systems are managed with the minimum of disruption to the users of the data architecture.

0154 The data architecture must support a versioning strategy for all its components which minimises coupling between data suppliers and data consumers and maximises both forward and backward compatibility.

Appendix 1 – Sample XML schemas

Basic data types

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="uk.rail.rcminterchange/0.1/"
xmlns="uk.rail.rcminterchange/0.1/" xmlns:xs="http://www.w3.org/
2001/XMLSchema" xmlns:ukrrcm="uk.rail.rcminterchange/0.1/" ele-
mentFormDefault="qualified">
	<xs:simpleType name="ValueClass">
		<xs:restriction base="xs:string">
			<xs:enumeration value="BinaryData"/>
			<xs:enumeration value="BinaryObject"/>
			<xs:enumeration value="Boolean"/>
			<xs:enumeration value="Coordinate"/>
			<xs:enumeration value="EnumerationItem"/>
			<xs:enumeration value="Measure"/>
			<xs:enumeration value="MonetaryAmount"/>
			<xs:enumeration value="Number"/>
			<xs:enumeration value="Percentage"/>
			<xs:enumeration value="Probability"/>
			<xs:enumeration value="Text"/>
			<xs:enumeration value="TextWithUnit"/>
			<xs:enumeration value="URI"/>
			<xs:enumeration value="UTCDateTime"/>
			<xs:enumeration value="UUID"/>
			<xs:enumeration value="XMLAny"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="navigability">
		<xs:restriction base="xs:string">
			<xs:enumeration value="AB"/>
			<xs:enumeration value="BA"/>
			<xs:enumeration value="both"/>
			<xs:enumeration value="none"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="startEndFlag">
		<xs:annotation>
			<xs:documentation>0 indicates the start of
the segment, 1 indicates the end</xs:documentation>
		</xs:annotation>
		<xs:restriction base="xs:int">
			<xs:enumeration value="0"/>
			<xs:enumeration value="1"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="ELR">
```

```
            <xs:annotation>
                  <xs:documentation>ELR must be a valid Engi-
neers' Line Reference (3 or 4-char alphanumeric string)</xs:doc-
umentation>
            </xs:annotation>
            <xs:list itemType="xs:string"/>
      </xs:simpleType>
      <xs:simpleType name="TIPLOC">
            <xs:list itemType="xs:string"/>
      </xs:simpleType>
      <xs:simpleType name="LengthKM">
            <xs:annotation>
                  <xs:documentation>km and fractional km in
the form kkkk.mmm</xs:documentation>
            </xs:annotation>
            <xs:list itemType="xs:decimal"/>
      </xs:simpleType>
      <xs:simpleType name="TrackID">
            <xs:annotation>
                  <xs:documentation>Track ID must be a valid
numeric track ID using standard nomenclature such as 2100 or
1200</xs:documentation>
            </xs:annotation>
            <xs:list itemType="xs:string"/>
      </xs:simpleType>
      <xs:simpleType name="RunningLine">
            <xs:list itemType="xs:string"/>
      </xs:simpleType>
      <xs:simpleType name="UUID">
            <xs:annotation>
                  <xs:documentation>UUID is a GUID of 32 bytes
represented as hex characters with dashes. Format xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxx (8-4-4-4-12 characters). See RFC 4122 - A
Universally Unique IDentifier (UUID) URN Namespace.</xs:documen-
tation>
            </xs:annotation>
            <xs:restriction base="xs:string"/>
      </xs:simpleType>
      <xs:element name="ValueContent" type="ValueContent"/>
      <xs:complexType name="ValueContent">
            <xs:sequence>
                  <xs:choice minOccurs="1" maxOccurs="1">
                        <xs:element name="BinaryData"
type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="BinaryObject"
type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="Boolean"
type="xs:boolean" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="Coordinate"
type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="EnumerationItem"
type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="Measure"
type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="MonetaryAmount"
type="xs:string" minOccurs="1" maxOccurs="1"/>
```

```
                        <xs:element name="Number"
type="xs:double" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="Percentage"
type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="Probability"
type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="Text"
type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="TextWithUnit"
type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="URI"
type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="UTCDateTime"
type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="UUID"
type="xs:string" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="XMLAny"
type="xs:string" minOccurs="1" maxOccurs="1"/>
                    </xs:choice>
                </xs:sequence>
        </xs:complexType>
        <xs:simpleType name="SegmentTypeEnum">
                <xs:annotation>
                        <xs:documentation>Descriptive term for seg-
ment type</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Axle"/>
                        <xs:enumeration value="Bearing"/>
                        <xs:enumeration value="Bogie"/>
                        <xs:enumeration value="Crossover"/>
                        <xs:enumeration value="OLE"/>
                        <xs:enumeration value="Rail"/>
                        <xs:enumeration value="Route Section"/>
                        <xs:enumeration value="Suspension"/>
                        <xs:enumeration value="Switch"/>
                        <xs:enumeration value="Track Section"/>
                        <xs:enumeration value="Vehicle End"/>
                        <xs:enumeration value="Wheel"/>
                        <xs:enumeration value="Wheelset"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="SegmentType">
                <xs:annotation>
                        <xs:documentation>Descriptive term for the
data type. Enumerated list in SegmentTypeEnum are preferred val-
ues.</xs:documentation>
                </xs:annotation>
                <xs:union memberTypes="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="DistanceMilesChains">
                <xs:annotation>
                        <xs:documentation>miles and chains in the
form mmm:cc. The miles figure mmm need not be zero-padded; the
chains cc must be padded to the left. cc must be in the range 00
to 79</xs:documentation>
```

```
            </xs:annotation>
            <xs:list itemType="xs:string"/>
      </xs:simpleType>
      <xs:simpleType name="DistanceKM">
            <xs:annotation>
                  <xs:documentation>km and fractional km in
the form kkkk.mmm</xs:documentation>
            </xs:annotation>
            <xs:restriction base="xs:decimal"/>
      </xs:simpleType>
</xs:schema>
```

Extensions to MIMOSA core data model

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="uk.rail.rcminterchange/0.1/"
xmlns="uk.rail.rcminterchange/0.1/" xmlns:xs="http://www.w3.org/
2001/XMLSchema" xmlns:osacbm="http://www.mimosa.org/OSACB-
MV3.3.1">
      <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="DataManipulation.xsd"/>
      <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="DataAcquisition.xsd"/>
      <xs:element name="DMURI" type="DMURI"/>
      <xs:complexType name="DMURI">
            <xs:annotation>
                  <xs:documentation>URI to Data Manipulation
data.</xs:documentation>
            </xs:annotation>
            <xs:complexContent>
                  <xs:extension base="osacbm:DMDataEvent">
                        <xs:sequence>
                              <xs:element name="ContentType"
type="osacbm:Mime" minOccurs="0" maxOccurs="1"/>
                              <xs:element name="URI"
type="xs:anyURI" minOccurs="1" maxOccurs="1"/>
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
      <xs:element name="DAURI" type="DAURI"/>
      <xs:complexType name="DAURI">
            <xs:annotation>
                  <xs:documentation>URI to raw acquired
data.</xs:documentation>
            </xs:annotation>
            <xs:complexContent>
                  <xs:extension base="osacbm:DADataEvent">
                        <xs:sequence>
                              <xs:element name="ContentType"
type="osacbm:Mime" minOccurs="1" maxOccurs="1"/>
                              <xs:element name="URI"
type="xs:anyURI" minOccurs="1" maxOccurs="1"/>
```

```
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
        </xs:complexType>
</xs:schema>
```

Datagram format

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="uk.rail.rcminterchange/0.1/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="uk.rail.rcminterchange/0.1/"
xmlns:ukrrcm="uk.rail.rcminterchange/0.1/" xmlns:osacbm="http://
www.mimosa.org/OSACBMV3.3.1" xmlns:dcterms="http://purl.org/dc/
terms/" elementFormDefault="qualified">

       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="StateDetection.xsd"/>
       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="PrognosticsAssessment.xsd"/>
       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="MonitorIdGroup.xsd"/>
       <xs:import namespace="http://purl.org/dc/terms/" sche-
maLocation="dcterms.xsd"/>
       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="HealthAssessment.xsd"/>
       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="DataManipulation.xsd"/>
       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="DataAcquisition.xsd"/>
       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="Configuration.xsd"/>
       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="ConfigRequest.xsd"/>
       <xs:import namespace="http://www.mimosa.org/OSACB-
MV3.3.1" schemaLocation="AdvisoryGeneration.xsd"/>
       <xs:include schemaLocation="ukrrcmcore.xsd"/>
       <xs:element name="body" type="ukrrcm:body"/>
       <xs:complexType name="body">
              <xs:sequence minOccurs="0" maxOccurs="unbounded">
                     <xs:element name="AGDataEvent"
type="osacbm:AGDataEvent" minOccurs="0" maxOccurs="unbounded"/>
                     <xs:element name="AGPort"
type="osacbm:AGPort" minOccurs="0" maxOccurs="unbounded"/>
                     <xs:element name="ConfigRequest"
type="osacbm:ConfigRequest" minOccurs="0" maxOccurs="unbounded"/
>
                     <xs:element name="Configuration"
type="osacbm:Configuration" minOccurs="0" maxOccurs="unbounded"/
```

```
>
                    <xs:element name="DADataEvent"
type="osacbm:DADataEvent" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="DAPort"
type="osacbm:DAPort" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="DMDataEvent"
type="osacbm:DMDataEvent" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="DMPort"
type="osacbm:DMPort" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="Entity"
type="ukrrcm:Entity" minOccurs="0" maxOccurs="unbounded">
                           <xs:annotation>
                                  <xs:documentation>Entities
representing segments and assets</xs:documentation>
                           </xs:annotation>
                    </xs:element>
                    <xs:element name="HADataEvent"
type="osacbm:HADataEvent" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="HAPort"
type="osacbm:HAPort" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="metadata"
type="dcterms:elementOrRefinementContainer" minOccurs="0" maxOc-
curs="1">
                           <xs:annotation>
                                  <xs:documentation>Container
for Qualified Dublin Core Type metadata elements</xs:documenta-
tion>
                           </xs:annotation>
                    </xs:element>
                    <xs:element name="MonitorIdGroup"
type="osacbm:MonitorIdGroup" minOccurs="0" maxOc-
curs="unbounded"/>
                    <xs:element name="MonitorIdGroupList"
type="osacbm:MonitorIdGroupList" minOccurs="0" maxOc-
curs="unbounded"/>
                    <xs:element name="PADataEvent"
type="osacbm:PADataEvent" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="PAPort"
type="osacbm:PAPort" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="SDDataEvent"
type="osacbm:SDDataEvent" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="SDPort" type="xs:string"
minOccurs="0" maxOccurs="unbounded"/>
                    <xs:any namespace="##other" minOccurs="0"
maxOccurs="unbounded">
                           <xs:annotation>
                                  <xs:documentation>Provide an
extension point for custom application specific elements</
xs:documentation>
                           </xs:annotation>
                    </xs:any>
             </xs:sequence>
      </xs:complexType>
      <xs:element name="header" type="ukrrcm:header"/>
      <xs:complexType name="header">
             <xs:annotation>
```

```
                        <xs:documentation>Header for a UK Rail RCM
datagram. Required once for each message.</xs:documentation>
                </xs:annotation>
                <xs:sequence minOccurs="1" maxOccurs="1">
                        <xs:element ref="dcterms:created" minOc-
curs="1" maxOccurs="1">
                                <xs:annotation>
                                        <xs:documentation>Timestamp of
the message in ISO 8601 format</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element ref="dcterms:format" minOc-
curs="0" maxOccurs="1">
                                <xs:annotation>
                                        <xs:documentation>Format of the
data contained within the message</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element ref="dcterms:license" minOc-
curs="0" maxOccurs="1"/>
                        <xs:element ref="dcterms:rights" minOc-
curs="0" maxOccurs="1"/>
                        <xs:element ref="dcterms:rightsHolder" min-
Occurs="0" maxOccurs="1"/>
                        <xs:element ref="dcterms:source" minOc-
curs="1" maxOccurs="1">
                                <xs:annotation>
                                        <xs:documentation>Source of the
data.</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element name="version" type="xs:string"
minOccurs="0" maxOccurs="1">
                                <xs:annotation>
                                        <xs:documentation>Version of
the architecture to which the message is compliant</xs:documen-
tation>
                                </xs:annotation>
                        </xs:element>
                        <xs:any namespace="##other" minOccurs="0"
maxOccurs="unbounded">
                                <xs:annotation>
                                        <xs:documentation>Provide an
extension point for custom application specific elements</
xs:documentation>
                                </xs:annotation>
                        </xs:any>
                </xs:sequence>
        </xs:complexType>
        <xs:element name="rcmMessage" type="ukrrcm:rcmMessage"/>
        <xs:complexType name="rcmMessage">
                <xs:sequence>
                        <xs:element name="header"
type="ukrrcm:header" minOccurs="1" maxOccurs="1"/>
                        <xs:element name="body" type="ukrrcm:body"
minOccurs="0" maxOccurs="1"/>
```

```
            </xs:sequence>
      </xs:complexType>
</xs:schema>
```

Network topology

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="uk.rail.rcminterchange/0.1/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="uk.rail.rcminterchange/0.1/"
xmlns:ukrrcm="uk.rail.rcminterchange/0.1/" elementFormDe-
fault="qualified">
      <xs:include schemaLocation="ukrrcmtype.xsd"/>
      <xs:include schemaLocation="ukrrcmcore.xsd"/>
      <xs:element name="PublicLeg" type="ukrrcm:PublicLeg"/>
      <xs:complexType name="PublicLeg">
            <xs:annotation>
                  <xs:documentation>Attribute to contain
entity that describes station to station train leg</xs:documen-
tation>
            </xs:annotation>
            <xs:complexContent>
                  <xs:extension base="ukrrcm:Attribute">
                        <xs:sequence>
                              <xs:element name="JourneyLeg"
type="ukrrcm:JourneyLeg" minOccurs="1" maxOccurs="1"/>
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
      <xs:element name="JourneyLeg" type="ukrrcm:JourneyLeg"/>
      <xs:complexType name="JourneyLeg">
            <xs:annotation>
                  <xs:documentation>Entity to hold attributes
that describe a link between two stations between which a passen-
ger train can travel</xs:documentation>
            </xs:annotation>
            <xs:complexContent>
                  <xs:extension base="ukrrcm:AttributableEn-
tity">
                        <xs:sequence>
                              <xs:element name="EndStation"
type="ukrrcm:Station" minOccurs="1" maxOccurs="1"/>
                              <xs:element name="StartSta-
tion" type="ukrrcm:Station" minOccurs="1" maxOccurs="1"/>
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
      <xs:element name="NetworkLink" type="ukrrcm:NetworkLink"/
>
      <xs:complexType name="NetworkLink">
```

```
            <xs:annotation>
                  <xs:documentation>Entity to hold attributes
that describe a link between two timing points along which a
train can travel</xs:documentation>
            </xs:annotation>
            <xs:complexContent>
                  <xs:extension base="ukrrcm:AttributableEn-
tity">
                        <xs:sequence>
                              <xs:element name="LengthKM"
type="ukrrcm:LengthKM" minOccurs="1" maxOccurs="1"/>
                              <xs:element name="StartTIPLOC"
type="ukrrcm:TIPLOC" minOccurs="1" maxOccurs="1"/>
                              <xs:element name="EndTIPLOC"
type="ukrrcm:TIPLOC" minOccurs="1" maxOccurs="1"/>
                              <xs:element name="RunningLine"
type="ukrrcm:RunningLine" minOccurs="0" maxOccurs="1"/>
                              <xs:element name="TrackID"
type="ukrrcm:TrackID" minOccurs="0" maxOccurs="1"/>
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
      <xs:element name="SegmentPublicLeg" type="ukrrcm:Segment-
PublicLeg"/>
      <xs:complexType name="SegmentPublicLeg">
            <xs:annotation>
                  <xs:documentation>Represents station to sta-
tion link i.e. public train leg</xs:documentation>
            </xs:annotation>
            <xs:complexContent>
                  <xs:extension base="ukrrcm:Segment">
                        <xs:sequence>
                              <xs:element name="PublicLeg"
type="ukrrcm:PublicLeg" minOccurs="1" maxOccurs="1"/>
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
      <xs:element name="TimingLeg" type="ukrrcm:TimingLeg"/>
      <xs:complexType name="TimingLeg">
            <xs:annotation>
                  <xs:documentation>Attribute to contain
entity that describes a timing leg for a train</xs:documenta-
tion>
            </xs:annotation>
            <xs:complexContent>
                  <xs:extension base="ukrrcm:Attribute">
                        <xs:sequence>
                              <xs:element name="NetworkLink"
type="ukrrcm:NetworkLink" minOccurs="1" maxOccurs="1"/>
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
      <xs:element name="SegmentTiming" type="ukrrcm:SegmentTim-
```

```
ing"/>
	<xs:complexType name="SegmentTiming">
		<xs:annotation>
			<xs:documentation>Represents a link between
two timing points along which a train can travel</xs:documenta-
tion>
		</xs:annotation>
		<xs:complexContent>
			<xs:extension base="ukrrcm:Segment">
				<xs:sequence>
					<xs:element name="TimingLeg"
type="ukrrcm:TimingLeg" minOccurs="1" maxOccurs="1"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:element name="Station" type="ukrrcm:Station"/>
	<xs:complexType name="Station">
		<xs:complexContent>
			<xs:extension base="ukrrcm:Attribute">
				<xs:sequence/>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
</xs:schema>
```

Rolling stock

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="uk.rail.rcminterchange/0.1/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="uk.rail.rcminterchange/0.1/"
xmlns:ukrrcm="uk.rail.rcminterchange/0.1/" elementFormDe-
fault="qualified">
	<xs:include schemaLocation="ukrrcmcore.xsd"/>
	<xs:element name="Formation" type="ukrrcm:Formation"/>
	<xs:complexType name="Formation">
		<xs:complexContent>
			<xs:extension base="ukrrcm:Segment">
				<xs:sequence>
					<xs:element name="Vehicle"
type="ukrrcm:Vehicle" minOccurs="0" maxOccurs="unbounded"/>
					<xs:element name="UnitNumber"
type="ukrrcm:UnitNumber" minOccurs="0" maxOccurs="1"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:element name="Vehicle" type="ukrrcm:Vehicle"/>
	<xs:complexType name="Vehicle">
		<xs:complexContent>
			<xs:extension base="ukrrcm:Segment">
				<xs:sequence>
```

```
                                    <xs:element name="EVN"
type="ukrrcm:EVN" minOccurs="0" maxOccurs="1"/>
                                    <xs:element name="PaintedNum-
ber" type="ukrrcm:PaintedNumber" minOccurs="0" maxOccurs="1"/>
                                    <xs:element name="VehicleSta-
tus" type="ukrrcm:VehicleStatus" minOccurs="0" maxOccurs="1"/>
                                    <xs:element name="Vehicle-
Class" type="ukrrcm:VehicleClass" minOccurs="0" maxOccurs="1"/>
                                    <xs:element name="Vehicle-
Owner" type="ukrrcm:VehicleOwner" minOccurs="0" maxOccurs="1"/>
                                    <xs:element name="VehicleMain-
tainer" type="ukrrcm:VehicleMaintainer" minOccurs="0" maxOc-
curs="1"/>
                                    <xs:element name="VehicleEnd"
type="ukrrcm:VehicleEnd" minOccurs="0" maxOccurs="2"/>
                                    <xs:element name="VehiclePosi-
tion" type="ukrrcm:VehiclePosition" minOccurs="0" maxOccurs="1"/
>
                                    <xs:element name="Pantograph"
type="ukrrcm:Pantograph" minOccurs="0" maxOccurs="unbounded"/>
                                    <xs:element name="Orientation"
type="ukrrcm:Orientation" minOccurs="0" maxOccurs="1"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="Pantograph" type="ukrrcm:Pantograph"/>
        <xs:complexType name="Pantograph">
                <xs:complexContent>
                        <xs:extension base="ukrrcm:Segment">
                                <xs:sequence>
                                    <xs:element name="EnergyMeter"
type="ukrrcm:EnergyMeter" minOccurs="0" maxOccurs="unbounded"/>
                                    <xs:element name="Pantograph-
Geometry" type="ukrrcm:PantographGeometry" minOccurs="0" maxOc-
curs="1"/>
                                    <xs:element name="Pantograph-
Type" type="ukrrcm:PantographType" minOccurs="0" maxOccurs="1"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:simpleType name="EVN">
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="PaintedNumber">
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="VehicleStatus">
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="VehicleClass">
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="VehicleOwner">
                <xs:list itemType="xs:string"/>
```

```
	</xs:simpleType>
	<xs:simpleType name="UnitNumber">
		<xs:list itemType="xs:string"/>
	</xs:simpleType>
	<xs:element name="VehicleEnd" type="ukrrcm:VehicleEnd"/>
	<xs:complexType name="VehicleEnd">
		<xs:complexContent>
			<xs:extension base="ukrrcm:Segment">
				<xs:sequence>
					<xs:element name="DriversCab"
type="ukrrcm:DriversCab" minOccurs="0" maxOccurs="1"/>
					<xs:element name="Coupling"
type="ukrrcm:Coupling" minOccurs="0" maxOccurs="1"/>
					<xs:element name="Bogie"
type="ukrrcm:Bogie" minOccurs="0" maxOccurs="unbounded"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:element name="DriversCab" type="ukrrcm:DriversCab"/>
	<xs:complexType name="DriversCab">
		<xs:complexContent>
			<xs:extension base="ukrrcm:Segment">
				<xs:sequence>
					<xs:element name="Radio"
type="ukrrcm:Radio" minOccurs="0" maxOccurs="unbounded"/>
					<xs:element name="ETCS"
type="ukrrcm:ETCS" minOccurs="0" maxOccurs="1"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:element name="Radio" type="ukrrcm:Radio"/>
	<xs:complexType name="Radio">
		<xs:complexContent>
			<xs:extension base="ukrrcm:Segment">
				<xs:sequence>
					<xs:element name="NRNRadioNum-
ber" type="ukrrcm:NRNRadioNumber" minOccurs="0" maxOccurs="1"/>
					<xs:element name="CSRRadioNum-
ber" type="ukrrcm:CSRRadioNumber" minOccurs="0" maxOccurs="1"/>
					<xs:element name="RETBNumber"
type="ukrrcm:RETBNumber" minOccurs="0" maxOccurs="1"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:element name="Coupling" type="ukrrcm:Coupling"/>
	<xs:complexType name="Coupling">
		<xs:complexContent>
			<xs:extension base="ukrrcm:Segment">
				<xs:sequence>
					<xs:element name="Coupling-
Type" type="ukrrcm:CouplingType" minOccurs="0" maxOccurs="1"/>
				</xs:sequence>
			</xs:extension>
```

```
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="ETCS" type="ukrrcm:ETCS"/>
        <xs:complexType name="ETCS">
                <xs:complexContent>
                        <xs:extension base="ukrrcm:Segment">
                                <xs:sequence/>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:simpleType name="NRNRadioNumber">
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="CSRRadioNumber">
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="VehicleMaintainer">
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:element name="Bogie" type="ukrrcm:Bogie"/>
        <xs:complexType name="Bogie">
                <xs:complexContent>
                        <xs:extension base="ukrrcm:Segment">
                                <xs:sequence>
                                        <xs:element name="Wheelset-
Mounting" type="ukrrcm:WheelsetMounting" minOccurs="0" maxOc-
curs="unbounded"/>
                                        <xs:element name="Suspension"
type="ukrrcm:Suspension" minOccurs="0" maxOccurs="unbounded"/>
                                        <xs:element name="BogieFrame"
type="ukrrcm:BogieFrame" minOccurs="0" maxOccurs="1"/>
                                        <xs:element name="Wheelbase"
type="ukrrcm:Wheelbase" minOccurs="0" maxOccurs="1"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="Wheelset" type="ukrrcm:Wheelset"/>
        <xs:complexType name="Wheelset">
                <xs:complexContent>
                        <xs:extension base="ukrrcm:Segment">
                                <xs:sequence>
                                        <xs:element name="Axle"
type="ukrrcm:Axle" minOccurs="0" maxOccurs="unbounded"/>
                                        <xs:element name="Wheel"
type="ukrrcm:Wheel" minOccurs="0" maxOccurs="unbounded"/>
                                        <xs:element name="Wheelset-
Guage" type="ukrrcm:WheelsetGuage" minOccurs="0" maxOccurs="1"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="Axle" type="ukrrcm:Axle"/>
        <xs:complexType name="Axle">
                <xs:complexContent>
                        <xs:extension base="ukrrcm:Segment">
```

```
                        <xs:sequence/>
                    </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="Wheel" type="ukrrcm:Wheel"/>
        <xs:complexType name="Wheel">
                <xs:complexContent>
                    <xs:extension base="ukrrcm:Segment">
                        <xs:sequence>
                            <xs:element name="WheelDiame-
ter" type="ukrrcm:WheelDiameter" minOccurs="0" maxOccurs="1"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="WheelsetMounting" type="ukrrcm:Wheel-
setMounting"/>
        <xs:complexType name="WheelsetMounting">
                <xs:complexContent>
                    <xs:extension base="ukrrcm:Segment">
                        <xs:sequence>
                            <xs:element name="Bearing"
type="ukrrcm:Bearing" minOccurs="0" maxOccurs="unbounded"/>
                            <xs:element name="Wheelset"
type="ukrrcm:Wheelset" minOccurs="0" maxOccurs="unbounded"/>
                            <xs:element name="Suspension"
type="ukrrcm:Suspension" minOccurs="0" maxOccurs="unbounded"/>
                            <xs:element name="BrakeSystem"
type="ukrrcm:BrakeSystem" minOccurs="0" maxOccurs="unbounded"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="Bearing" type="ukrrcm:Bearing"/>
        <xs:complexType name="Bearing">
                <xs:complexContent>
                    <xs:extension base="ukrrcm:Segment">
                        <xs:sequence/>
                    </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="Suspension" type="ukrrcm:Suspension"/>
        <xs:complexType name="Suspension">
                <xs:complexContent>
                    <xs:extension base="ukrrcm:Segment">
                        <xs:sequence/>
                    </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="EnergyMeter" type="ukrrcm:EnergyMeter"/
>
        <xs:complexType name="EnergyMeter">
                <xs:complexContent>
                    <xs:extension base="ukrrcm:Segment">
                        <xs:sequence/>
                    </xs:extension>
```

```
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="BogieFrame" type="ukrrcm:BogieFrame"/>
        <xs:complexType name="BogieFrame">
                <xs:complexContent>
                        <xs:extension base="ukrrcm:Segment">
                                <xs:sequence/>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:simpleType name="VehiclePosition">
                <xs:annotation>
                        <xs:documentation>Position of the vehicle
within its formation. Integer which increments one end of forma-
tion to other.</xs:documentation>
                </xs:annotation>
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="Orientation">
                <xs:annotation>
                        <xs:documentation>Forward or Reverse. For-
ward if end "1" is towards the front of the formation.</xs:docu-
mentation>
                </xs:annotation>
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="RETBNumber">
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:element name="BrakeSystem" type="ukrrcm:BrakeSystem"/
>
        <xs:complexType name="BrakeSystem">
                <xs:complexContent>
                        <xs:extension base="ukrrcm:Segment">
                                <xs:sequence>
                                        <xs:element name="BrakeType"
type="ukrrcm:BrakeType" minOccurs="0" maxOccurs="1"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:simpleType name="BrakeType">
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="CouplingType">
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="Wheelbase">
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="PantographType">
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="PantographGeometry">
                <xs:list itemType="xs:string"/>
        </xs:simpleType>
```

```
	<xs:simpleType name="WheelsetGuage">
		<xs:list itemType="xs:string"/>
	</xs:simpleType>
	<xs:simpleType name="WheelDiameter">
		<xs:list itemType="xs:string"/>
	</xs:simpleType>
</xs:schema>
```

Fixed point asset

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="uk.rail.rcminterchange/0.1/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="uk.rail.rcminterchange/0.1/"
xmlns:ukrrcm="uk.rail.rcminterchange/0.1/" elementFormDe-
fault="qualified">
	<xs:include schemaLocation="ukrrcmtype.xsd"/>
	<xs:include schemaLocation="ukrrcmcore.xsd"/>
	<xs:element name="AssetFixedPoint" type="ukrrcm:Asset-
FixedPoint"/>
	<xs:complexType name="AssetFixedPoint">
		<xs:annotation>
			<xs:documentation>Represents an asset (com-
ponent) with a fixed point location such as point heater.</
xs:documentation>
		</xs:annotation>
		<xs:complexContent>
			<xs:extension base="ukrrcm:Asset">
				<xs:sequence>
					<xs:choice minOccurs="0" maxOc-
curs="1">
						<xs:annotation>
							<xs:documentation>Attri-
bute to describe the position of a point location asset in terms
of ELR and mileage e.g. switch</xs:documentation>
						</xs:annotation>
						<xs:element name="Fixed-
PointLocationKM" type="ukrrcm:FixedPointLocationKM" minOc-
curs="1" maxOccurs="1"/>
						<xs:element name="Fixed-
PointLocationMilesChains" type="ukrrcm:FixedPointLocation-
MilesChains" minOccurs="1" maxOccurs="1"/>
					</xs:choice>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:element name="FixedPointLocationMilesChains"
type="ukrrcm:FixedPointLocationMilesChains"/>
	<xs:complexType name="FixedPointLocationMilesChains">
		<xs:annotation>
			<xs:documentation>Entity to hold attributes
```

```
that describe the position of a point location asset in terms of
ELR and distance from datum in miles and chains</xs:documenta-
tion>
            </xs:annotation>
            <xs:complexContent>
                  <xs:extension base="ukrrcm:AttributableEn-
tity">
                        <xs:sequence>
                              <xs:element name="Distance-
MilesChains" type="ukrrcm:DistanceMilesChains" minOccurs="1"
maxOccurs="1"/>
                              <xs:element name="ELR"
type="ukrrcm:ELR" minOccurs="1" maxOccurs="1"/>
                              <xs:element name="TrackID"
type="ukrrcm:TrackID" minOccurs="0" maxOccurs="1"/>
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
      <xs:element name="FixedPointLocationKM"
type="ukrrcm:FixedPointLocationKM"/>
      <xs:complexType name="FixedPointLocationKM">
            <xs:annotation>
                  <xs:documentation>Entity to hold attributes
that describe the position of a point location asset in terms of
ELR and distance from datum in kilometers</xs:documentation>
            </xs:annotation>
            <xs:complexContent>
                  <xs:extension base="ukrrcm:AttributableEn-
tity">
                        <xs:sequence>
                              <xs:element name="DistanceKM"
type="ukrrcm:DistanceKM" minOccurs="1" maxOccurs="1"/>
                              <xs:element name="ELR"
type="ukrrcm:ELR" minOccurs="1" maxOccurs="1"/>
                              <xs:element name="TrackID"
type="ukrrcm:TrackID" minOccurs="0" maxOccurs="1"/>
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
      <xs:element name="SegmentFixedPoint" type="ukrrcm:Seg-
mentFixedPoint"/>
      <xs:complexType name="SegmentFixedPoint">
            <xs:annotation>
                  <xs:documentation>Represents a segment with
a fixed position on the network such as a switch.</xs:documenta-
tion>
            </xs:annotation>
            <xs:complexContent>
                  <xs:extension base="ukrrcm:Segment">
                        <xs:sequence>
                              <xs:choice minOccurs="0" maxOc-
curs="1">
                                    <xs:annotation>
                                          <xs:documentation>Attri-
```

```
bute to describe the position of a point location asset in terms
of ELR and mileage e.g. switch</xs:documentation>
                                        </xs:annotation>
                                        <xs:element name="Fixed-
PointLocationKM" type="ukrrcm:FixedPointLocationKM" minOc-
curs="1" maxOccurs="1"/>
                                        <xs:element name="Fixed-
PointLocationMilesChains" type="ukrrcm:FixedPointLocation-
MilesChains" minOccurs="1" maxOccurs="1"/>
                                    </xs:choice>
                                    <xs:element name="AssetFixed-
Point" type="ukrrcm:AssetFixedPoint" minOccurs="0" maxOc-
curs="1"/>
                            </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
</xs:schema>
```

Fixed linear asset

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="uk.rail.rcminterchange/0.1/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="uk.rail.rcminterchange/0.1/"
xmlns:ukrrcm="uk.rail.rcminterchange/0.1/" elementFormDe-
fault="qualified">
        <xs:include schemaLocation="ukrrcmtype.xsd"/>
        <xs:include schemaLocation="ukrrcmcore.xsd"/>
        <xs:element name="AssetLinear" type="ukrrcm:AssetLinear"/
>
        <xs:complexType name="AssetLinear">
                <xs:annotation>
                        <xs:documentation>Represents a linear asset
(component) such as a specific piece of rail.</xs:documentation>
                </xs:annotation>
                <xs:complexContent>
                        <xs:extension base="ukrrcm:Asset">
                                <xs:sequence>
                                        <xs:choice minOccurs="0" maxOc-
curs="1">
                                                <xs:annotation>
                                                        <xs:documentation>Ele-
ment that describes a the position of fixed linear location e.g.
segment of track</xs:documentation>
                                                </xs:annotation>
                                                <xs:element name="Fixed-
LinearLocationKM" type="ukrrcm:FixedLinearLocationKM" minOc-
curs="1" maxOccurs="1"/>
                                                <xs:element name="Fixed-
LinearLocationMilesChains" type="ukrrcm:FixedLinearLocation-
```

```
MilesChains" minOccurs="1" maxOccurs="1"/>
                                    </xs:choice>
                              </xs:sequence>
                        </xs:extension>
                  </xs:complexContent>
            </xs:complexType>
            <xs:element name="SegmentLinear" type="ukrrcm:SegmentLin-
ear"/>
            <xs:complexType name="SegmentLinear">
                  <xs:annotation>
                        <xs:documentation>Represents a segment with
a fixed linear position</xs:documentation>
                  </xs:annotation>
                  <xs:complexContent>
                        <xs:extension base="ukrrcm:Segment">
                              <xs:sequence>
                                    <xs:choice minOccurs="1" maxOc-
curs="1">
                                          <xs:annotation>
                                                <xs:documentation>Ele-
ment that describes a the position of fixed linear location e.g.
segment of track</xs:documentation>
                                          </xs:annotation>
                                          <xs:element name="Fixed-
LinearLocationKM" type="ukrrcm:FixedLinearLocationKM" minOc-
curs="1" maxOccurs="1"/>
                                          <xs:element name="Fixed-
LinearLocationMilesChains" type="ukrrcm:FixedLinearLocation-
MilesChains" minOccurs="1" maxOccurs="1"/>
                                    </xs:choice>
                                    <xs:element name="AssetLinear"
type="ukrrcm:AssetLinear" minOccurs="0" maxOccurs="unbounded"/>
                              </xs:sequence>
                        </xs:extension>
                  </xs:complexContent>
            </xs:complexType>
            <xs:element name="FixedLinearLocationMilesChains"
type="ukrrcm:FixedLinearLocationMilesChains"/>
            <xs:complexType name="FixedLinearLocationMilesChains">
                  <xs:annotation>
                        <xs:documentation>Entity to hold attributes
that describe the position of a linear asset e.g. track using
miles and chains distance from the datum</xs:documentation>
                  </xs:annotation>
                  <xs:complexContent>
                        <xs:extension base="ukrrcm:AttributableEn-
tity">
                              <xs:sequence>
                                    <xs:element name="StartDistan-
ceMiles" type="ukrrcm:DistanceMilesChains" minOccurs="1" maxOc-
curs="1"/>
                                    <xs:element name="ELR"
type="ukrrcm:ELR" minOccurs="1" maxOccurs="1"/>
                                    <xs:element name="EndDistance-
Miles" type="ukrrcm:DistanceMilesChains" minOccurs="1" maxOc-
curs="1"/>
```

```
                                    <xs:element name="TrackID"
type="ukrrcm:TrackID" minOccurs="0" maxOccurs="1"/>
                              </xs:sequence>
                        </xs:extension>
                  </xs:complexContent>
            </xs:complexType>
            <xs:element name="FixedLinearLocationKM"
type="ukrrcm:FixedLinearLocationKM"/>
            <xs:complexType name="FixedLinearLocationKM">
                  <xs:annotation>
                        <xs:documentation>Entity to hold attributes
that describe the position of a linear asset e.g. track using
kilometers distance from datum</xs:documentation>
                  </xs:annotation>
                  <xs:complexContent>
                        <xs:extension base="ukrrcm:AttributableEn-
tity">
                              <xs:sequence>
                                    <xs:element name="ELR"
type="ukrrcm:ELR" minOccurs="1" maxOccurs="1"/>
                                    <xs:element name="TrackID"
type="ukrrcm:TrackID" minOccurs="0" maxOccurs="1"/>
                                    <xs:element name="EndDis-
tanceKM" type="ukrrcm:DistanceKM" minOccurs="1" maxOccurs="1"/>
                                    <xs:element name="StartDis-
tanceKM" type="ukrrcm:DistanceKM" minOccurs="1" maxOccurs="1"/>
                              </xs:sequence>
                        </xs:extension>
                  </xs:complexContent>
            </xs:complexType>
</xs:schema>
```

Appendix 2 – Sample datagrams

XML datagrams

These datagrams are expressed in XML using the schemas described in Section Example XML Schemas. Whilst very verbose, this format can be verified automatically against the published schema and can accept user-defined extensions without causing validation failures.

More succinct text-based datagrams are shown below.

Asset-related RCM data

This sample datagram shows peak current data for a number of point motors attached to switches in a crossover. It includes the position of the switches as well as GUIDs for them, their component parts and the whole crossover of which they are part.

```
<?xml version="1.0" encoding="utf-8"?>
<rcmMessage xmlns="uk.rail.rcminterchange/0.1/" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xmlns:mim="http://www.mimosa.org/
OSACBMV3.3.1">
  <header>
    <created>2014-09-23T09:20:42.343</created>
    <format>XML</format>
    <license>anyType</license>
    <rights>restricted</rights>
    <rightsHolder>Network Rail</rightsHolder>
    <source>ORBIS</source>
    <version>0.1</version>
  </header>
  <body>
    <Entity xsi:type="SegmentFixedPoint">
      <GUID>AA92C515-E67D-4356-ADBC-20615A44196A</GUID>
      <Name>Netley Crossover</Name>
      <SegmentType>Crossover</SegmentType>
      <Segment xsi:type="SegmentFixedPoint">
        <GUID>53F5E86C-DB48-4DEA-A1B6-2B7353483702</GUID>
        <Name>Netley Up Switch</Name>
        <SegmentType>Switch</SegmentType>
        <AssetFixedPoint>
          <GUID>D1B0139B-D748-41A5-AA82-E86E1F84EB45</GUID>
          <metadata>1</metadata>
          <Name>Netley Crossover Up Switch Motor</Name>
          <mim:MeasLoc >
```

```
            <measLocId xmlns="">
              <id>1029535</id>
              <site>
                <category>SITE_SPECIFIC</category>
                <systemUserTag>ORBIS</systemUserTag>
                <userTag>ORBIS</userTag>
              </site>
            </measLocId>
            <measLocType xmlns="">
              <code>9823598</code>
              <dbId>0</dbId>
              <name>Switch Motor Current RCM</name>
              <site>
                <category>SITE_SPECIFIC</category>
                <systemUserTag>ORBIS</systemUserTag>
                <userTag>ORBIS</userTag>
              </site>
            </measLocType>
           <name xmlns="">Netley Crossover Up Switch Motor peak current</
name>
            <userTag xmlns="">Netley Crossover Up Switch Motor peak cur-
rent</userTag>
          </mim:MeasLoc>
          <mim:DataEvent xsi:type="mim:DMReal">
            <id xmlns="">1029535</id>
            <site xmlns="">
              <category>SITE_SPECIFIC</category>
              <systemUserTag>ORBIS</systemUserTag>
              <userTag>ORBIS</userTag>
            </site>
            <time xmlns="">
              <time>2014-09-23T08:20:12.343</time>
              <time_type>OSACBM_TIME_MIMOSA</time_type>
            </time>
            <dataStatus xmlns="">OK</dataStatus>
            <value xmlns="">1.24</value>
          </mim:DataEvent>
          <mim:DataEvent xsi:type="mim:DMReal">
            <id xmlns="">1029535</id>
            <site xmlns="">
              <category>SITE_SPECIFIC</category>
              <systemUserTag>ORBIS</systemUserTag>
              <userTag>ORBIS</userTag>
            </site>
            <time xmlns="">
              <time>2014-09-23T08:20:24.923</time>
              <time_type>OSACBM_TIME_MIMOSA</time_type>
            </time>
            <dataStatus xmlns="">OK</dataStatus>
            <value xmlns="">1.12</value>
          </mim:DataEvent>
          <mim:DataEvent xsi:type="mim:DMReal">
            <id xmlns="">1029535</id>
            <site xmlns="">
              <category>SITE_SPECIFIC</category>
```

```
            <systemUserTag>ORBIS</systemUserTag>
            <userTag>ORBIS</userTag>
          </site>
          <time xmlns="">
            <time>2014-09-23T08:20:36.523</time>
            <time_type>OSACBM_TIME_MIMOSA</time_type>
          </time>
          <dataStatus xmlns="">OK</dataStatus>
          <value xmlns="">1.29</value>
        </mim:DataEvent>
        <AssetType>Switch Motor</AssetType>
        <SerialNumber>1085-2564</SerialNumber>
        <FixedPointLocationMilesChains>
          <DistanceMilesChains>6:42</DistanceMilesChains>
          <ELR >LSW</ELR>
          <TrackID>1100</TrackID>
        </FixedPointLocationMilesChains>
      </AssetFixedPoint>
    </Segment>
    <Segment xsi:type="SegmentFixedPoint">
      <GUID>00D4AD3D-A4B1-437C-B857-65648D40815E</GUID>
      <Name>Netley Down Switch</Name>
      <SegmentType>Switch</SegmentType>
      <AssetFixedPoint>
        <GUID>199BA73D-2215-40E7-AB60-5EF911A8D5D6</GUID>
        <metadata>1</metadata>
        <Name>Netley Crossover Down Switch Motor</Name>
        <MeasLoc xmlns="http://www.mimosa.org/OSACBMV3.3.1">
          <measLocId xmlns="">
            <id>1029564</id>
            <site>
              <category>SITE_SPECIFIC</category>
              <systemUserTag>ORBIS</systemUserTag>
              <userTag>ORBIS</userTag>
            </site>
          </measLocId>
          <measLocType xmlns="">
            <code>9823598</code>
            <dbId>0</dbId>
            <name>Switch Motor Current RCM</name>
            <site>
              <category>SITE_SPECIFIC</category>
              <systemUserTag>ORBIS</systemUserTag>
              <userTag>ORBIS</userTag>
            </site>
          </measLocType>
          <name xmlns="">Netley Crossover Down Switch Motor peak cur-
rent</name>
           <userTag xmlns="">Netley Crossover Donw Switch Motor peak cur-
rent</userTag>
        </MeasLoc>
        <DataEvent xsi:type="DMReal" xmlns="http://www.mimosa.org/
OSACBMV3.3.1">
          <id xmlns="">1029564</id>
          <site xmlns="">
```

```
            <category>SITE_SPECIFIC</category>
            <systemUserTag>ORBIS</systemUserTag>
            <userTag>ORBIS</userTag>
          </site>
          <time xmlns="">
            <time>2014-09-23T08:20:14.568</time>
            <time_type>OSACBM_TIME_MIMOSA</time_type>
          </time>
          <dataStatus xmlns="">OK</dataStatus>
          <value xmlns="">1.19</value>
        </DataEvent>
        <DataEvent xsi:type="DMReal" xmlns="http://www.mimosa.org/
OSACBMV3.3.1">
          <id xmlns="">1029564</id>
          <site xmlns="">
            <category>SITE_SPECIFIC</category>
            <systemUserTag>ORBIS</systemUserTag>
            <userTag>ORBIS</userTag>
          </site>
          <time xmlns="">
            <time>2014-09-23T08:20:24.101</time>
            <time_type>OSACBM_TIME_MIMOSA</time_type>
          </time>
          <dataStatus xmlns="">OK</dataStatus>
          <value xmlns="">1.23</value>
        </DataEvent>
        <DataEvent xsi:type="DMReal" xmlns="http://www.mimosa.org/
OSACBMV3.3.1">
          <id xmlns="">1029564</id>
          <site xmlns="">
            <category>SITE_SPECIFIC</category>
            <systemUserTag>ORBIS</systemUserTag>
            <userTag>ORBIS</userTag>
          </site>
          <time xmlns="">
            <time>2014-09-23T08:20:39.567</time>
            <time_type>OSACBM_TIME_MIMOSA</time_type>
          </time>
          <dataStatus xmlns="">OK</dataStatus>
          <value xmlns="">1.21</value>
        </DataEvent>
        <AssetType>Switch Motor</AssetType>
        <SerialNumber>1085-256</SerialNumber>
        <FixedPointLocationMilesChains>
          <DistanceMilesChains>6:42</DistanceMilesChains>
          <ELR >LSW</ELR>
          <TrackID>2100</TrackID>
        </FixedPointLocationMilesChains>
      </AssetFixedPoint>
    </Segment>
    <FixedPointLocationMilesChains>
      <DistanceMilesChains>6:42</DistanceMilesChains>
      <ELR >LSW</ELR>
    </FixedPointLocationMilesChains>
  </Entity>
```

```
  </body>
</rcmMessage>
```

Moving asset data

The sample XML datagrams in this section show a request / response pair for the most recent information available about some bearings on a specific rail vehicle.

Bearing data request

This datagram shows how data might be requested from a system which has bearing RCM data using a MIMOSA standard query pattern.

```
<?xml version="1.0" encoding="utf-8"?>
<rcmMessage xmlns="uk.rail.rcminterchange/0.1/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mim="http://www.mimosa.org/OSACBMV3.3.1">
  <header>
    <created>2014-09-23T09:43:22.986Z</created>
    <format>XML</format>
    <source>TOC RS Mgt System</source>
    <version>0.1</version>
  </header>
  <!--In this sample message we will requesting the recent data
relating
  to bearing fault detection for a couple of bearings that have
had
  poor health assessments.-->
  <body>
    <Entity xsi:type="Bearing">
      <GUID>B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1</GUID>
      <Name>Class 350 vehicle 50401 End 1 Bogie wheelset 1 side
A</Name>
      <Tag>50401-1-1A</Tag>
      <mim:TimeSelect>
        <!--We could specifiy a time range here but will specifiy
         the last 3 alert event sets in this example-->
         <snapshotStart xmlns="">3</snapshotStart>
      </mim:TimeSelect>
    </Entity>
    <Entity xsi:type="Bearing">
      <GUID>B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1</GUID>
      <Name>Class 350 vehicle 50401 End 2 Bogie wheelset 1 side
A</Name>
      <Tag>50401-2-1A</Tag>
      <mim:TimeSelect>
        <!--We could specifiy a time range here but will specifiy
         the last 3 alert event sets in this example-->
         <snapshotStart xmlns="">3</snapshotStart>
      </mim:TimeSelect>
    </Entity>
  </body>
```

```
</rcmMessage>
```

Bearing data response

This datagram shows the sort of XML-encoded response an RCM data system might return in response to the query above. It includes both State Detection and Health Assessment data as well as embedded links to the raw sound files captured by an acoustic bearing monitoring system.

```
<?xml version="1.0" encoding="utf-8"?>
<rcmMessage xmlns="uk.rail.rcminterchange/0.1/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mim="http://www.mimosa.org/OSACBMV3.3.1"
xmlns:ukrrcm="uk.rail.rcminterchange/0.1/">
  <header>
    <created>2014-09-23T09:49:26.485Z</created>
    <format>XML</format>
    <license>restricted</license>
    <rights>restricted</rights>
    <rightsHolder>Bearing Monitoing System Owner</rightsHolder>
    <source>Bearing Monitoing System</source>
    <version>0.1</version>
  </header>
  <!--In this sample message we will requesting the recent data
relating
  to bearing fault detection for a couple of bearings that have
had
  poor health assessments.-->
  <body>
    <Entity xsi:type="Bearing">
      <GUID>B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1</GUID>
      <Name>Class 350 vehicle 50401 End 1 Bogie wheelset 1 side
A</Name>
      <Tag>50401-1-1A</Tag>
      <!--Our first data event set includes state detection mes-
sage that
      indicates no problems for this bearing. URL to the RCM
audio
      file is included as a DM message-->
      <mim:DataEventSet>
        <alertStatus xmlns="">true</alertStatus>
        <dataEvents xsi:type="mim:SDEnumSet" xmlns="">
          <alertStatus xmlns="">false</alertStatus>
          <id xmlns="">771111111</id>
          <sequenceNum xmlns="">0</sequenceNum>
          <site xmlns="">
            <category>SITE_ENT_ZERO</category>
            <userTag>UK Rail RCM</userTag>
          </site>
          <dataStatus xmlns="">OK</dataStatus>
          <values xmlns="">
            <value>
```

```
        <name>CUP</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>CONE</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>ETCH</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>SPUN</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>GROWLER</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>ROLLER</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>LBR</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>UNKNOWN</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>MULTIPLE</name>
        <value>0</value>
      </value>
    </values>
  </dataEvents>
  <dataEvents xsi:type="ukrrcm:DMURI" xmlns="">
    <id xmlns="">771111111</id>
    <site xmlns="">
```

```
            <category>SITE_ENT_ZERO</category>
            <userTag>UK Rail RCM</userTag>
          </site>
          <dataStatus xmlns="">OK</dataStatus>
          <ContentType xmlns="">
            <value>audio/mpeg3</value>
          </ContentType>
          <URI xmlns="">http://nrdatahost.co.uk/bearing/
fhkd8762k9d.wav</URI>
        </dataEvents>
        <id xmlns="">771111111</id>
        <sequenceNum xmlns="">459735</sequenceNum>
        <site xmlns="">
          <category>SITE_ENT_ZERO</category>
          <userTag>UK Rail RCM</userTag>
        </site>
        <time xmlns="">
          <time>2014-09-22T06:41:24.923Z</time>
          <time_type>OSACBM_TIME_MIMOSA</time_type>
        </time>
      </mim:DataEventSet>
      <!--Our second data event set includes state detection mes-
sage that
      indicates a problem for this bearing. URL to the RCM audio
      file is included as a DM message-->
      <mim:DataEventSet>
        <alertStatus xmlns="">true</alertStatus>
        <dataEvents xsi:type="mim:SDEnumSet" xmlns="">
          <alertStatus xmlns="">true</alertStatus>
          <id xmlns="">771111111</id>
          <sequenceNum xmlns="">0</sequenceNum>
          <site xmlns="">
            <category>SITE_ENT_ZERO</category>
            <userTag>UK Rail RCM</userTag>
          </site>
          <dataStatus xmlns="">OK</dataStatus>
          <numAlerts xmlns="">
            <alertName>Bearing Problem</alertName>
            <alertSeverity>
              <code>8459</code>
              <dbId>0</dbId>
              <name>Medium</name>
              <site>
                <category>SITE_ENT_ZERO</category>
                <userTag>UK Rail RCM</userTag>
              </site>
            </alertSeverity>
            <alertTypeCode>2971</alertTypeCode>
            <alertTypeId>0</alertTypeId>
            <alertTypeSite>
              <category>SITE_ENT_ZERO</category>
              <userTag>UK Rail RCM</userTag>
            </alertTypeSite>
          </numAlerts>
          <values xmlns="">
            <value>
```

```
        <name>CUP</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>CONE</name>
        <value>1</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>ETCH</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>SPUN</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>GROWLER</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>ROLLER</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>LBR</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>UNKNOWN</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>MULTIPLE</name>
        <value>0</value>
      </value>
    </values>
  </dataEvents>
  <dataEvents xsi:type="ukrrcm:DMURI" xmlns="">
    <id xmlns="">771111111</id>
    <site xmlns="">
```

```
            <category>SITE_ENT_ZERO</category>
            <userTag>UK Rail RCM</userTag>
          </site>
          <dataStatus xmlns="">OK</dataStatus>
          <ContentType xmlns="">
            <value>audio/mpeg3</value>
          </ContentType>
          <URI xmlns="">http://nrdatahost.co.uk/bearing/
fhkd8342k9d.wav</URI>
        </dataEvents>
        <id xmlns="">771111111</id>
        <sequenceNum xmlns="">459735</sequenceNum>
        <site xmlns="">
          <category>SITE_ENT_ZERO</category>
          <userTag>UK Rail RCM</userTag>
        </site>
        <time xmlns="">
          <time>2014-09-22T22:04:21.901Z</time>
          <time_type>OSACBM_TIME_MIMOSA</time_type>
        </time>
      </mim:DataEventSet>
      <!--Our third data event set includes state detection mes-
sage that
      indicates no problems for this bearing suggesting the sec-
ond message may
      have just been a suprious message hence just the 80% score
on the health
      assessment. URL to the RCM audio file is included as a DM
message-->
      <mim:DataEventSet>
        <alertStatus xmlns="">true</alertStatus>
        <dataEvents xsi:type="mim:SDEnumSet" xmlns="">
          <alertStatus xmlns="">true</alertStatus>
          <id xmlns="">771111111</id>
          <sequenceNum xmlns="">0</sequenceNum>
          <site xmlns="">
            <category>SITE_ENT_ZERO</category>
            <userTag>UK Rail RCM</userTag>
          </site>
          <dataStatus xmlns="">OK</dataStatus>
          <values xmlns="">
            <value>
              <name>CUP</name>
              <value>0</value>
            </value>
          </values>
          <values xmlns="">
            <value>
              <name>CONE</name>
              <value>0</value>
            </value>
          </values>
          <values xmlns="">
            <value>
              <name>ETCH</name>
              <value>0</value>
```

```
            </value>
          </values>
          <values xmlns="">
            <value>
              <name>SPUN</name>
              <value>0</value>
            </value>
          </values>
          <values xmlns="">
            <value>
              <name>GROWLER</name>
              <value>0</value>
            </value>
          </values>
          <values xmlns="">
            <value>
              <name>ROLLER</name>
              <value>0</value>
            </value>
          </values>
          <values xmlns="">
            <value>
              <name>LBR</name>
              <value>0</value>
            </value>
          </values>
          <values xmlns="">
            <value>
              <name>UNKNOWN</name>
              <value>0</value>
            </value>
          </values>
          <values xmlns="">
            <value>
              <name>MULTIPLE</name>
              <value>0</value>
            </value>
          </values>
        </dataEvents>
        <dataEvents xsi:type="ukrrcm:DMURI" xmlns="">
          <id xmlns="">771111111</id>
          <site xmlns="">
            <category>SITE_ENT_ZERO</category>
            <userTag>UK Rail RCM</userTag>
          </site>
          <dataStatus xmlns="">OK</dataStatus>
          <ContentType xmlns="">
            <value>audio/mpeg3</value>
          </ContentType>
          <URI xmlns="">http://nrdatahost.co.uk/bearing/
fhkd876df9d.wav</URI>
        </dataEvents>
        <id xmlns="">771111111</id>
        <sequenceNum xmlns="">459735</sequenceNum>
        <site xmlns="">
          <category>SITE_ENT_ZERO</category>
```

```
          <userTag>UK Rail RCM</userTag>
        </site>
        <time xmlns="">
          <time>2014-09-23T06:38:58.121Z</time>
          <time_type>OSACBM_TIME_MIMOSA</time_type>
        </time>
      </mim:DataEventSet>
    </Entity>
    <Entity xsi:type="Bearing">
      <GUID>B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1</GUID>
      <Name>Class 350 vehicle 50401 End 2 Bogie wheelset 1 side
A</Name>
      <Tag>50401-2-1A</Tag>
      <!--all the data event sets includes state detection mes-
sage that
      indicates multiple problems for this bearing hence the
health
      assessment score of 23.
      URL to the RCM audio files are included as DM messages-->
      <mim:DataEventSet>
        <alertStatus xmlns="">true</alertStatus>
        <dataEvents xsi:type="mim:SDEnumSet" xmlns="">
          <alertStatus xmlns="">false</alertStatus>
          <id xmlns="">771111111</id>
          <sequenceNum xmlns="">0</sequenceNum>
          <site xmlns="">
            <category>SITE_ENT_ZERO</category>
            <userTag>UK Rail RCM</userTag>
          </site>
          <dataStatus xmlns="">OK</dataStatus>
          <numAlerts xmlns="">
            <alertName>Bearing Problem</alertName>
            <alertSeverity>
              <code>8459</code>
              <dbId>0</dbId>
              <name>Medium</name>
              <site>
                <category>SITE_ENT_ZERO</category>
                <userTag>UK Rail RCM</userTag>
              </site>
            </alertSeverity>
            <alertTypeCode>2971</alertTypeCode>
            <alertTypeId>0</alertTypeId>
            <alertTypeSite>
              <category>SITE_ENT_ZERO</category>
              <userTag>UK Rail RCM</userTag>
            </alertTypeSite>
          </numAlerts>
          <values xmlns="">
            <value>
              <name>CUP</name>
              <value>0</value>
            </value>
          </values>
          <values xmlns="">
            <value>
```

```
        <name>CONE</name>
        <value>1</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>ETCH</name>
        <value>1</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>SPUN</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>GROWLER</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>ROLLER</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>LBR</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>UNKNOWN</name>
        <value>1</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>MULTIPLE</name>
        <value>1</value>
      </value>
    </values>
  </dataEvents>
  <dataEvents xsi:type="ukrrcm:DMURI" xmlns="">
    <id xmlns="">771111111</id>
    <site xmlns="">
      <category>SITE_ENT_ZERO</category>
      <userTag>UK Rail RCM</userTag>
    </site>
    <dataStatus xmlns="">OK</dataStatus>
    <ContentType xmlns="">
      <value>audio/mpeg3</value>
```

```
          </ContentType>
          <URI xmlns="">http://nrdatahost.co.uk/bearing/
fhkd8762k9d.wav</URI>
        </dataEvents>
        <id xmlns="">771111111</id>
        <sequenceNum xmlns="">459735</sequenceNum>
        <site xmlns="">
          <category>SITE_ENT_ZERO</category>
          <userTag>UK Rail RCM</userTag>
        </site>
        <time xmlns="">
          <time>2014-09-22T06:41:26.101Z</time>
          <time_type>OSACBM_TIME_MIMOSA</time_type>
        </time>
      </mim:DataEventSet>
      <mim:DataEventSet>
        <alertStatus xmlns="">true</alertStatus>
        <dataEvents xsi:type="mim:SDEnumSet" xmlns="">
          <alertStatus xmlns="">false</alertStatus>
          <id xmlns="">771111111</id>
          <sequenceNum xmlns="">0</sequenceNum>
          <site xmlns="">
            <category>SITE_ENT_ZERO</category>
            <userTag>UK Rail RCM</userTag>
          </site>
          <dataStatus xmlns="">OK</dataStatus>
          <numAlerts xmlns="">
            <alertName>Bearing Problem</alertName>
            <alertSeverity>
              <code>8459</code>
              <dbId>0</dbId>
              <name>Medium</name>
              <site>
                <category>SITE_ENT_ZERO</category>
                <userTag>UK Rail RCM</userTag>
              </site>
            </alertSeverity>
            <alertTypeCode>2971</alertTypeCode>
            <alertTypeId>0</alertTypeId>
            <alertTypeSite>
              <category>SITE_ENT_ZERO</category>
              <userTag>UK Rail RCM</userTag>
            </alertTypeSite>
          </numAlerts>
          <values xmlns="">
            <value>
              <name>CUP</name>
              <value>0</value>
            </value>
          </values>
          <values xmlns="">
            <value>
              <name>CONE</name>
              <value>1</value>
            </value>
          </values>
```

```
            <values xmlns="">
              <value>
                <name>ETCH</name>
                <value>1</value>
              </value>
            </values>
            <values xmlns="">
              <value>
                <name>SPUN</name>
                <value>0</value>
              </value>
            </values>
            <values xmlns="">
              <value>
                <name>GROWLER</name>
                <value>0</value>
              </value>
            </values>
            <values xmlns="">
              <value>
                <name>ROLLER</name>
                <value>0</value>
              </value>
            </values>
            <values xmlns="">
              <value>
                <name>LBR</name>
                <value>0</value>
              </value>
            </values>
            <values xmlns="">
              <value>
                <name>UNKNOWN</name>
                <value>1</value>
              </value>
            </values>
            <values xmlns="">
              <value>
                <name>MULTIPLE</name>
                <value>1</value>
              </value>
            </values>
          </dataEvents>
          <dataEvents xsi:type="ukrrcm:DMURI" xmlns="">
            <id xmlns="">771111111</id>
            <site xmlns="">
              <category>SITE_ENT_ZERO</category>
              <userTag>UK Rail RCM</userTag>
            </site>
            <dataStatus xmlns="">OK</dataStatus>
            <ContentType xmlns="">
              <value>audio/mpeg3</value>
            </ContentType>
            <URI xmlns="">http://nrdatahost.co.uk/bearing/
fhkd8342k9d.wav</URI>
          </dataEvents>
```

```
  <id xmlns="">771111111</id>
  <sequenceNum xmlns="">459735</sequenceNum>
  <site xmlns="">
    <category>SITE_ENT_ZERO</category>
    <userTag>UK Rail RCM</userTag>
  </site>
  <time xmlns="">
    <time>2014-09-22T22:04:23.611Z</time>
    <time_type>OSACBM_TIME_MIMOSA</time_type>
  </time>
</mim:DataEventSet>
<mim:DataEventSet>
  <alertStatus xmlns="">true</alertStatus>
  <dataEvents xsi:type="mim:SDEnumSet" xmlns="">
    <alertStatus xmlns="">false</alertStatus>
    <id xmlns="">771111111</id>
    <sequenceNum xmlns="">0</sequenceNum>
    <site xmlns="">
      <category>SITE_ENT_ZERO</category>
      <userTag>UK Rail RCM</userTag>
    </site>
    <dataStatus xmlns="">OK</dataStatus>
    <numAlerts xmlns="">
      <alertName>Bearing Problem</alertName>
      <alertSeverity>
        <code>8459</code>
        <dbId>0</dbId>
        <name>Medium</name>
        <site>
          <category>SITE_ENT_ZERO</category>
          <userTag>UK Rail RCM</userTag>
        </site>
      </alertSeverity>
      <alertTypeCode>2971</alertTypeCode>
      <alertTypeId>0</alertTypeId>
      <alertTypeSite>
        <category>SITE_ENT_ZERO</category>
        <userTag>UK Rail RCM</userTag>
      </alertTypeSite>
    </numAlerts>
    <values xmlns="">
      <value>
        <name>CUP</name>
        <value>0</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>CONE</name>
        <value>1</value>
      </value>
    </values>
    <values xmlns="">
      <value>
        <name>ETCH</name>
        <value>1</value>
```

```
            </value>
          </values>
          <values xmlns="">
            <value>
              <name>SPUN</name>
              <value>0</value>
            </value>
          </values>
          <values xmlns="">
            <value>
              <name>GROWLER</name>
              <value>0</value>
            </value>
          </values>
          <values xmlns="">
            <value>
              <name>ROLLER</name>
              <value>0</value>
            </value>
          </values>
          <values xmlns="">
            <value>
              <name>LBR</name>
              <value>0</value>
            </value>
          </values>
          <values xmlns="">
            <value>
              <name>UNKNOWN</name>
              <value>1</value>
            </value>
          </values>
          <values xmlns="">
            <value>
              <name>MULTIPLE</name>
              <value>1</value>
            </value>
          </values>
        </dataEvents>
        <dataEvents xsi:type="ukrrcm:DMURI" xmlns="">
          <id xmlns="">771111111</id>
          <site xmlns="">
            <category>SITE_ENT_ZERO</category>
            <userTag>UK Rail RCM</userTag>
          </site>
          <dataStatus xmlns="">OK</dataStatus>
          <ContentType xmlns="">
            <value>audio/mpeg3</value>
          </ContentType>
          <URI xmlns="">http://nrdatahost.co.uk/bearing/
fhkd876df9d.wav</URI>
        </dataEvents>
        <id xmlns="">771111111</id>
        <sequenceNum xmlns="">459735</sequenceNum>
        <site xmlns="">
          <category>SITE_ENT_ZERO</category>
```

```
          <userTag>UK Rail RCM</userTag>
        </site>
        <time xmlns="">
          <time>2014-09-23T06:39:00.238Z</time>
          <time_type>OSACBM_TIME_MIMOSA</time_type>
        </time>
      </mim:DataEventSet>
    </Entity>
  </body>
</rcmMessage>
```

Formation data

The datagrams in this section represent a request / response pair that might be interchanged between an RCM data user and a system responsible for assessing the health of a train formation.

Health request for a diesel multiple unit

This request asks for the current health assessment for a specific DMU.

```
<?xml version="1.0" encoding="utf-8"?>
<rcmMessage xmlns="uk.rail.rcminterchange/0.1/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mim="http://www.mimosa.org/OSACBMV3.3.1">
  <header>
    <created>2014-09-23T08:20:22.986Z</created>
    <format>XML</format>
    <source>TOC RS Mgt System</source>
    <version>0.1</version>
  </header>
  <!--In this sample message we will request the health assess-
ment
  for unit 170401 relating to Bearing Problems
  No time is specified so it will just return the Health Assess-
ment
  last known health assesment-->
  <body>
    <Entity xsi:type="Formation">
      <GUID>E7B34708-F040-4D99-92BE-D633BABA12AE</GUID>
      <metadata>1</metadata>
      <Name>170401</Name>
      <Tag>170401</Tag>
      <Order>1</Order>
      <SelectionFilter xsi:type="mim:AgentAlertFilter">
        <eventType xmlns="">
          <code>416</code>
          <dbId>1</dbId>
          <name>Bearing Problem</name>
          <site>
            <category>SITE_ENT_ZERO</category>
            <userTag>UK Rail RCM</userTag>
```

```
          </site>
        </eventType>
      </SelectionFilter>
      </Entity>
  </body>
</rcmMessage>
```

Response – health assessment detail

This response gives a complete dump of all the components of the requested DMU, with health assessment data for the whole unit and its components: vehicles, bogies, wheelsets, bearings.

```
<?xml version="1.0" encoding="utf-8"?>
<rcmMessage xmlns="uk.rail.rcminterchange/0.1/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mim="http://www.mimosa.org/OSACBMV3.3.1">
  <header>
    <created>2014-09-23T08:20:25.125Z</created>
    <format>XML</format>
    <license>restricted</license>
    <rights>restricted</rights>
    <rightsHolder>Bearing Monitoing System Owner</rightsHolder>
    <source>Bearing Monitoing System</source>
    <version>0.1</version>
  </header>
  <body>
    <Entity xsi:type="Formation">
      <GUID>E7B34708-F040-4D99-92BE-D633BABA12AE</GUID>
      <Name>Unit 170401</Name>
      <Tag>170401</Tag>
      <Order>1</Order>
      <DataEvent xsi:type="mim:HADataEvent">
        <!--This is the health assessment for the formation as a
whole-->
        <id xmlns="">90111111</id>
        <site xmlns="">
          <category>SITE_ENT_ZERO</category>
          <userTag>UK Rail RCM</userTag>
        </site>
        <time xmlns="">
          <time>2014-09-23T08:20:24.923Z</time>
          <time_type>OSACBM_TIME_MIMOSA</time_type>
        </time>
        <healthGood xmlns="">false</healthGood>
        <itemHealth xmlns="">
          <hLevel>23</hLevel>
          <item_id>
            <code>90111111</code>
            <name>Unit 170401</name>
            <segOrAs>SEGMENT</segOrAs>
            <site>
              <category>SITE_ENT_ZERO</category>
              <userTag>UK Rail RCM</userTag>
```

```
      </site>
    </item_id>
    <likelihood>0.95</likelihood>
    <utc_health>
      <time>2014-09-23T08:20:24.923Z</time>
      <time_type>OSACBM_TIME_MIMOSA</time_type>
    </utc_health>
  </itemHealth>
</DataEvent>
<Vehicle>
  <GUID>DC42B5DE-9588-4CA7-AC48-E56A6CD00709</GUID>
  <Name>Class 170 vehicle 50401</Name>
  <Tag>50401</Tag>
  <DataEvent xsi:type="mim:HADataEvent">
    <!--This is the health assessment for the vehicle-->
    <id xmlns="">801111111</id>
    <site xmlns="">
      <category>SITE_ENT_ZERO</category>
      <userTag>UK Rail RCM</userTag>
    </site>
    <time xmlns="">
      <time>2014-09-23T08:20:24.923Z</time>
      <time_type>OSACBM_TIME_MIMOSA</time_type>
    </time>
    <healthGood xmlns="">false</healthGood>
    <itemHealth xmlns="">
      <hLevel>23</hLevel>
      <item_id>
        <code>801111111</code>
        <name>Class 170 vehicle 50401</name>
        <segOrAs>SEGMENT</segOrAs>
        <site>
          <category>SITE_ENT_ZERO</category>
          <userTag>UK Rail RCM</userTag>
        </site>
      </item_id>
      <likelihood>0.95</likelihood>
      <utc_health>
        <time>2014-09-23T08:20:24.923Z</time>
        <time_type>OSACBM_TIME_MIMOSA</time_type>
      </utc_health>
    </itemHealth>
  </DataEvent>
  <PaintedNumber>50401</PaintedNumber>
  <VehicleStatus>C</VehicleStatus>
  <VehicleOwner>HA</VehicleOwner>
  <VehicleMaintainer>Bombardier</VehicleMaintainer>
  <VehicleEnd>
    <GUID>8C7FC29E-9B42-4C30-B1DD-DA91B8421C97</GUID>
    <Name>Class 170 vehicle 50401 End 1</Name>
    <Tag>50401-1</Tag>
    <Bogie>
      <GUID>394CFEDF-F22D-4E9C-87AB-3CDE024663F6</GUID>
      <Name>Class 350 vehicle 50401 End 1 Bogie</Name>
      <Tag>50401-1</Tag>
      <DataEvent xsi:type="mim:HADataEvent">
```

```
                <!--This is the health assessment for the bogie-->
                <id xmlns="">79111111</id>
                <site xmlns="">
                  <category>SITE_ENT_ZERO</category>
                  <userTag>UK Rail RCM</userTag>
                </site>
                <time xmlns="">
                  <time>2014-09-23T08:20:24.923Z</time>
                  <time_type>OSACBM_TIME_MIMOSA</time_type>
                </time>
                <healthGood xmlns="">false</healthGood>
                <itemHealth xmlns="">
                  <hLevel>80</hLevel>
                  <item_id>
                    <code>79111111</code>
                   <name>Class 350 vehicle 50401 End 1 Bogie</name>
                    <segOrAs>SEGMENT</segOrAs>
                    <site>
                      <category>SITE_ENT_ZERO</category>
                      <userTag>UK Rail RCM</userTag>
                    </site>
                  </item_id>
                  <likelihood>0.95</likelihood>
                  <utc_health>
                    <time>2014-09-23T08:20:24.923Z</time>
                    <time_type>OSACBM_TIME_MIMOSA</time_type>
                  </utc_health>
                </itemHealth>
              </DataEvent>
              <WheelsetMounting>
                <GUID>EB81BA9B-EB36-4646-884F-CBCD449DD22A</GUID>
                <Name>Class 350 vehicle 50401 End 1 Bogie wheelset
1</Name>
                <Tag>50401-1-1</Tag>
                <DataEvent xsi:type="mim:HADataEvent">
                  <!--This is the health assessment for the wheel-
set-->
                  <id xmlns="">781111111</id>
                  <site xmlns="">
                    <category>SITE_ENT_ZERO</category>
                    <userTag>UK Rail RCM</userTag>
                  </site>
                  <time xmlns="">
                    <time>2014-09-23T08:20:24.923Z</time>
                    <time_type>OSACBM_TIME_MIMOSA</time_type>
                  </time>
                  <healthGood xmlns="">false</healthGood>
                  <itemHealth xmlns="">
                    <hLevel>80</hLevel>
                    <item_id>
                      <code>781111111</code>
                      <name>Class 350 vehicle 50401 End 1 Bogie
wheelset 1</name>
                      <segOrAs>SEGMENT</segOrAs>
                      <site>
                        <category>SITE_ENT_ZERO</category>
```

```
                    <userTag>UK Rail RCM</userTag>
                  </site>
                </item_id>
                <likelihood>0.95</likelihood>
                <utc_health>
                  <time>2014-09-23T08:20:24.923Z</time>
                  <time_type>OSACBM_TIME_MIMOSA</time_type>
                </utc_health>
              </itemHealth>
            </DataEvent>
            <Bearing>
             <GUID>B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1</GUID>
            <Name>Class 350 vehicle 50401 End 1 Bogie wheelset
1 side A</Name>
              <Tag>50401-1-1A</Tag>
              <DataEvent xsi:type="mim:HADataEvent">
               <!--This is the health assessment for the bear-
ing-->
               <id xmlns="">771111111</id>
               <site xmlns="">
                 <category>SITE_ENT_ZERO</category>
                 <userTag>UK Rail RCM</userTag>
               </site>
               <time xmlns="">
                 <time>2014-09-23T08:20:24.923Z</time>
                 <time_type>OSACBM_TIME_MIMOSA</time_type>
               </time>
               <healthGood xmlns="">false</healthGood>
               <itemHealth xmlns="">
                 <hLevel>80</hLevel>
                 <item_id>
                   <code>771111111</code>
                   <name>Class 350 vehicle 50401 End 1 Bogie
wheelset 1 side A</name>
                     <segOrAs>SEGMENT</segOrAs>
                     <site>
                       <category>SITE_ENT_ZERO</category>
                       <userTag>UK Rail RCM</userTag>
                     </site>
                   </item_id>
                   <likelihood>0.95</likelihood>
                   <utc_health>
                     <time>2014-09-23T08:20:24.923Z</time>
                     <time_type>OSACBM_TIME_MIMOSA</time_type>
                   </utc_health>
                 </itemHealth>
               </DataEvent>
             </Bearing>
             <Bearing>
              <GUID>B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1</GUID>
             <Name>Class 350 vehicle 50401 End 1 Bogie wheelset
1 side B</Name>
              <Tag>50401-1-1B</Tag>
              <DataEvent xsi:type="mim:HADataEvent">
               <!--This is the health assessment for the bear-
ing-->
```

```
                    <id xmlns="">771111112</id>
                    <site xmlns="">
                      <category>SITE_ENT_ZERO</category>
                      <userTag>UK Rail RCM</userTag>
                    </site>
                    <time xmlns="">
                      <time>2014-09-23T08:20:24.923Z</time>
                      <time_type>OSACBM_TIME_MIMOSA</time_type>
                    </time>
                    <healthGood xmlns="">false</healthGood>
                    <itemHealth xmlns="">
                      <hLevel>98</hLevel>
                      <item_id>
                        <code>771111112</code>
                        <name>Class 350 vehicle 50401 End 1 Bogie
wheelset 1 side B</name>
                        <segOrAs>SEGMENT</segOrAs>
                        <site>
                          <category>SITE_ENT_ZERO</category>
                          <userTag>UK Rail RCM</userTag>
                        </site>
                      </item_id>
                      <likelihood>0.95</likelihood>
                      <utc_health>
                        <time>2014-09-23T08:20:24.923Z</time>
                        <time_type>OSACBM_TIME_MIMOSA</time_type>
                      </utc_health>
                    </itemHealth>
                  </DataEvent>
                </Bearing>
              </WheelsetMounting>
              <WheelsetMounting>
                <GUID>EB81BA9B-EB36-4646-884F-CBCD449DD22A</GUID>
                <Name>Class 350 vehicle 50401 End 1 Bogie wheelset
2</Name>
                <Tag>50401-1-1</Tag>
                <DataEvent xsi:type="mim:HADataEvent">
                  <!--This is the health assessment for the wheel-
set-->
                  <id xmlns="">781111211</id>
                  <site xmlns="">
                    <category>SITE_ENT_ZERO</category>
                    <userTag>UK Rail RCM</userTag>
                  </site>
                  <time xmlns="">
                    <time>2014-09-23T08:20:24.923Z</time>
                    <time_type>OSACBM_TIME_MIMOSA</time_type>
                  </time>
                  <healthGood xmlns="">false</healthGood>
                  <itemHealth xmlns="">
                    <hLevel>96</hLevel>
                    <item_id>
                      <code>781111211</code>
                      <name>Class 350 vehicle 50401 End 1 Bogie
wheelset 2</name>
                      <segOrAs>SEGMENT</segOrAs>
```

```
                    <site>
                      <category>SITE_ENT_ZERO</category>
                      <userTag>UK Rail RCM</userTag>
                    </site>
                  </item_id>
                  <likelihood>0.95</likelihood>
                  <utc_health>
                    <time>2014-09-23T08:20:24.923Z</time>
                    <time_type>OSACBM_TIME_MIMOSA</time_type>
                  </utc_health>
                </itemHealth>
              </DataEvent>
              <Bearing>
               <GUID>B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1</GUID>
              <Name>Class 350 vehicle 50401 End 1 Bogie wheelset
2 side A</Name>
                <Tag>50401-1-1A</Tag>
                <DataEvent xsi:type="mim:HADataEvent">
                 <!--This is the health assessment for the bear-
ing-->
                  <id xmlns="">771111211</id>
                  <site xmlns="">
                    <category>SITE_ENT_ZERO</category>
                    <userTag>UK Rail RCM</userTag>
                  </site>
                  <time xmlns="">
                    <time>2014-09-23T08:20:24.923Z</time>
                    <time_type>OSACBM_TIME_MIMOSA</time_type>
                  </time>
                  <healthGood xmlns="">false</healthGood>
                  <itemHealth xmlns="">
                    <hLevel>96</hLevel>
                    <item_id>
                      <code>771111211</code>
                      <name>Class 350 vehicle 50401 End 1 Bogie
wheelset 2 side A</name>
                      <segOrAs>SEGMENT</segOrAs>
                      <site>
                        <category>SITE_ENT_ZERO</category>
                        <userTag>UK Rail RCM</userTag>
                      </site>
                    </item_id>
                    <likelihood>0.95</likelihood>
                    <utc_health>
                      <time>2014-09-23T08:20:24.923Z</time>
                      <time_type>OSACBM_TIME_MIMOSA</time_type>
                    </utc_health>
                  </itemHealth>
                </DataEvent>
              </Bearing>
              <Bearing>
               <GUID>B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1</GUID>
              <Name>Class 350 vehicle 50401 End 1 Bogie wheelset
2 side B</Name>
                <Tag>50401-1-1B</Tag>
                <DataEvent xsi:type="mim:HADataEvent">
```

```
                    <!--This is the health assessment for the bear-
ing-->
                     <id xmlns="">771111212</id>
                     <site xmlns="">
                       <category>SITE_ENT_ZERO</category>
                       <userTag>UK Rail RCM</userTag>
                     </site>
                     <time xmlns="">
                       <time>2014-09-23T08:20:24.923Z</time>
                       <time_type>OSACBM_TIME_MIMOSA</time_type>
                     </time>
                     <healthGood xmlns="">false</healthGood>
                     <itemHealth xmlns="">
                       <hLevel>97</hLevel>
                       <item_id>
                         <code>771111212</code>
                         <name>Class 350 vehicle 50401 End 1 Bogie
wheelset 2 side B</name>
                         <segOrAs>SEGMENT</segOrAs>
                         <site>
                           <category>SITE_ENT_ZERO</category>
                           <userTag>UK Rail RCM</userTag>
                         </site>
                       </item_id>
                       <likelihood>0.95</likelihood>
                       <utc_health>
                         <time>2014-09-23T08:20:24.923Z</time>
                         <time_type>OSACBM_TIME_MIMOSA</time_type>
                       </utc_health>
                     </itemHealth>
                   </DataEvent>
                 </Bearing>
               </WheelsetMounting>
             </Bogie>
           </VehicleEnd>
           <VehicleEnd>
             <GUID>8C7FC29E-9B42-4C30-B1DD-DA91B8421C97</GUID>
             <Name>Class 170 vehicle 50401 End 2</Name>
             <Tag>50401-2</Tag>
             <Bogie>
               <GUID>394CFEDF-F22D-4E9C-87AB-3CDE024663F6</GUID>
               <Name>Class 350 vehicle 50401 End 2 Bogie</Name>
               <Tag>50401-2</Tag>
               <DataEvent xsi:type="mim:HADataEvent">
                 <!--This is the health assessment for the bogie-->
                 <id xmlns="">79111121</id>
                 <site xmlns="">
                   <category>SITE_ENT_ZERO</category>
                   <userTag>UK Rail RCM</userTag>
                 </site>
                 <time xmlns="">
                   <time>2014-09-23T08:20:24.923Z</time>
                   <time_type>OSACBM_TIME_MIMOSA</time_type>
                 </time>
                 <healthGood xmlns="">false</healthGood>
                 <itemHealth xmlns="">
```

```
                    <hLevel>23</hLevel>
                    <item_id>
                      <code>79111121</code>
                     <name>Class 350 vehicle 50401 End 2 Bogie</name>
                      <segOrAs>SEGMENT</segOrAs>
                      <site>
                        <category>SITE_ENT_ZERO</category>
                        <userTag>UK Rail RCM</userTag>
                      </site>
                    </item_id>
                    <likelihood>0.95</likelihood>
                    <utc_health>
                      <time>2014-09-23T08:20:24.923Z</time>
                      <time_type>OSACBM_TIME_MIMOSA</time_type>
                    </utc_health>
                  </itemHealth>
                </DataEvent>
                <WheelsetMounting>
                  <GUID>EB81BA9B-EB36-4646-884F-CBCD449DD22A</GUID>
                  <Name>Class 350 vehicle 50401 End 2 Bogie wheelset
1</Name>
                  <Tag>50401-2-1</Tag>
                  <DataEvent xsi:type="mim:HADataEvent">
                    <!--This is the health assessment for the wheel-
set-->
                    <id xmlns="">781111121</id>
                    <site xmlns="">
                      <category>SITE_ENT_ZERO</category>
                      <userTag>UK Rail RCM</userTag>
                    </site>
                    <time xmlns="">
                      <time>2014-09-23T08:20:24.923Z</time>
                      <time_type>OSACBM_TIME_MIMOSA</time_type>
                    </time>
                    <healthGood xmlns="">false</healthGood>
                    <itemHealth xmlns="">
                      <hLevel>98</hLevel>
                      <item_id>
                        <code>781111121</code>
                        <name>Class 350 vehicle 50401 End 2 Bogie
wheelset 1</name>
                        <segOrAs>SEGMENT</segOrAs>
                        <site>
                          <category>SITE_ENT_ZERO</category>
                          <userTag>UK Rail RCM</userTag>
                        </site>
                      </item_id>
                      <likelihood>0.95</likelihood>
                      <utc_health>
                        <time>2014-09-23T08:20:24.923Z</time>
                        <time_type>OSACBM_TIME_MIMOSA</time_type>
                      </utc_health>
                    </itemHealth>
                  </DataEvent>
                  <Bearing>
                   <GUID>B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1</GUID>
```

```
              <Name>Class 350 vehicle 50401 End 2 Bogie wheelset
1 side A</Name>
                <Tag>50401-2-1A</Tag>
                <DataEvent xsi:type="mim:HADataEvent">
                 <!--This is the health assessment for the bear-
ing-->
                  <id xmlns="">771111121</id>
                  <site xmlns="">
                    <category>SITE_ENT_ZERO</category>
                    <userTag>UK Rail RCM</userTag>
                  </site>
                  <time xmlns="">
                    <time>2014-09-23T08:20:24.923Z</time>
                    <time_type>OSACBM_TIME_MIMOSA</time_type>
                  </time>
                  <healthGood xmlns="">false</healthGood>
                  <itemHealth xmlns="">
                    <hLevel>98</hLevel>
                    <item_id>
                      <code>771111121</code>
                      <name>Class 350 vehicle 50401 End 2 Bogie
wheelset 1 side A</name>
                      <segOrAs>SEGMENT</segOrAs>
                      <site>
                        <category>SITE_ENT_ZERO</category>
                        <userTag>UK Rail RCM</userTag>
                      </site>
                    </item_id>
                    <likelihood>0.95</likelihood>
                    <utc_health>
                      <time>2014-09-23T08:20:24.923Z</time>
                      <time_type>OSACBM_TIME_MIMOSA</time_type>
                    </utc_health>
                  </itemHealth>
                </DataEvent>
              </Bearing>
              <Bearing>
                <GUID>B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1</GUID>
              <Name>Class 350 vehicle 50401 End 2 Bogie wheelset
1 side B</Name>
                <Tag>50401-2-1B</Tag>
                <DataEvent xsi:type="mim:HADataEvent">
                 <!--This is the health assessment for the bear-
ing-->
                  <id xmlns="">771111122</id>
                  <site xmlns="">
                    <category>SITE_ENT_ZERO</category>
                    <userTag>UK Rail RCM</userTag>
                  </site>
                  <time xmlns="">
                    <time>2014-09-23T08:20:24.923Z</time>
                    <time_type>OSACBM_TIME_MIMOSA</time_type>
                  </time>
                  <healthGood xmlns="">false</healthGood>
                  <itemHealth xmlns="">
                    <hLevel>100</hLevel>
```

```
                    <item_id>
                      <code>771111122</code>
                      <name>Class 350 vehicle 50401 End 2 Bogie
wheelset 1 side B</name>
                      <segOrAs>SEGMENT</segOrAs>
                      <site>
                        <category>SITE_ENT_ZERO</category>
                        <userTag>UK Rail RCM</userTag>
                      </site>
                    </item_id>
                    <likelihood>0.95</likelihood>
                    <utc_health>
                      <time>2014-09-23T08:20:24.923Z</time>
                      <time_type>OSACBM_TIME_MIMOSA</time_type>
                    </utc_health>
                  </itemHealth>
                </DataEvent>
              </Bearing>
            </WheelsetMounting>
            <WheelsetMounting>
              <GUID>EB81BA9B-EB36-4646-884F-CBCD449DD22A</GUID>
              <Name>Class 350 vehicle 50401 End 2 Bogie wheelset
2</Name>
              <Tag>50401-2-1</Tag>
              <DataEvent xsi:type="mim:HADataEvent">
                <!--This is the health assessment for the wheel-
set-->
                <id xmlns="">781111221</id>
                <site xmlns="">
                  <category>SITE_ENT_ZERO</category>
                  <userTag>UK Rail RCM</userTag>
                </site>
                <time xmlns="">
                  <time>2014-09-23T08:20:24.923Z</time>
                  <time_type>OSACBM_TIME_MIMOSA</time_type>
                </time>
                <healthGood xmlns="">false</healthGood>
                <itemHealth xmlns="">
                  <hLevel>23</hLevel>
                  <item_id>
                    <code>781111221</code>
                    <name>Class 350 vehicle 50401 End 2 Bogie
wheelset 2</name>
                    <segOrAs>SEGMENT</segOrAs>
                    <site>
                      <category>SITE_ENT_ZERO</category>
                      <userTag>UK Rail RCM</userTag>
                    </site>
                  </item_id>
                  <likelihood>0.95</likelihood>
                  <utc_health>
                    <time>2014-09-23T08:20:24.923Z</time>
                    <time_type>OSACBM_TIME_MIMOSA</time_type>
                  </utc_health>
                </itemHealth>
              </DataEvent>
```

```
              <Bearing>
                <GUID>B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1</GUID>
              <Name>Class 350 vehicle 50401 End 2 Bogie wheelset
2 side A</Name>
                <Tag>50401-2-1A</Tag>
                <DataEvent xsi:type="mim:HADataEvent">
                 <!--This is the health assessment for the bear-
ing-->
                  <id xmlns="">771111221</id>
                  <site xmlns="">
                    <category>SITE_ENT_ZERO</category>
                    <userTag>UK Rail RCM</userTag>
                  </site>
                  <time xmlns="">
                    <time>2014-09-23T08:20:24.923Z</time>
                    <time_type>OSACBM_TIME_MIMOSA</time_type>
                  </time>
                  <healthGood xmlns="">false</healthGood>
                  <itemHealth xmlns="">
                    <hLevel>67</hLevel>
                    <item_id>
                      <code>771111221</code>
                      <name>Class 350 vehicle 50401 End 2 Bogie
wheelset 2 side A</name>
                      <segOrAs>SEGMENT</segOrAs>
                      <site>
                        <category>SITE_ENT_ZERO</category>
                        <userTag>UK Rail RCM</userTag>
                      </site>
                    </item_id>
                    <likelihood>0.95</likelihood>
                    <utc_health>
                      <time>2014-09-23T08:20:24.923Z</time>
                      <time_type>OSACBM_TIME_MIMOSA</time_type>
                    </utc_health>
                  </itemHealth>
                </DataEvent>
              </Bearing>
              <Bearing>
                <GUID>B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1</GUID>
              <Name>Class 350 vehicle 50401 End 2 Bogie wheelset
2 side B</Name>
                <Tag>50401-2-1B</Tag>
                <DataEvent xsi:type="mim:HADataEvent">
                 <!--This is the health assessment for the bear-
ing-->
                  <id xmlns="">771111222</id>
                  <site xmlns="">
                    <category>SITE_ENT_ZERO</category>
                    <userTag>UK Rail RCM</userTag>
                  </site>
                  <time xmlns="">
                    <time>2014-09-23T08:20:24.923Z</time>
                    <time_type>OSACBM_TIME_MIMOSA</time_type>
                  </time>
                  <healthGood xmlns="">false</healthGood>
```

```
                    <itemHealth xmlns="">
                      <hLevel>98</hLevel>
                      <item_id>
                        <code>771111222</code>
                        <name>Class 350 vehicle 50401 End 2 Bogie
wheelset 2 side B</name>
                        <segOrAs>SEGMENT</segOrAs>
                        <site>
                          <category>SITE_ENT_ZERO</category>
                          <userTag>UK Rail RCM</userTag>
                        </site>
                      </item_id>
                      <likelihood>0.95</likelihood>
                      <utc_health>
                        <time>2014-09-23T08:20:24.923Z</time>
                        <time_type>OSACBM_TIME_MIMOSA</time_type>
                      </utc_health>
                    </itemHealth>
                  </DataEvent>
                </Bearing>
              </WheelsetMounting>
            </Bogie>
          </VehicleEnd>
          <VehiclePosition></VehiclePosition>
          <Orientation></Orientation>
        </Vehicle>
        <Vehicle>
          <!--The other vehicles in the unit formation would be
here
          with all their data-->
        </Vehicle>
        <Vehicle>
          <!--The other vehicles in the unit formation would be
here
          with all their data-->
        </Vehicle>
      </Entity>
      <!--End Entity Formation-->
    </body>
</rcmMessage>
```

CSV datagrams

These sample datagrams are based on the same data as the XML ones above, but use the .csv text format which is more concise, more familiar to rail engineers and analysts and able to be imported directly into standard desktop software such as MS-Excel or MS-Access.

Bearing data response – Full version

This datagram shows all the data items present in the corresponding XML version in Section Bearing data response, but in .csv format.

```
created,format,license,rights,rightsHolder,source,version,BearingGUID,Bea
ringName,BearingTag,alertStatus,BearingMonitoringLocId,dataStatus,CUP,CON
E,ETCH,SPUN,GROWLER,ROLLER,LBR,UNKNOWN,MULTIPLE,ContentType,URI,alertType
Code,alertTypeId,alertTypeName,alertSeverityCode,alertSeveri-
tyName,sequenceNum,time
2014-09-23T09:49:26.485Z,CSV,restricted,restricted,Bearing Monitoring
System Owner,Bearing Monitoring System,0.1,4A77E289-F88A-49E0-B1D3-
F3C03B4C589B,Class 350 vehicle 50401 End 1 Bogie wheelset 1 side A,50401-
1-1A,FALSE,771111111,OK,0,0,0,0,0,0,0,0,audio/mpeg3,http://nrdata-
host.co.uk/bearing/fhkd8762h567.wav,,,,,,459735,2014-09-22T06:41:24.923Z
2014-09-23T09:49:26.485Z,CSV,restricted,restricted,Bearing Monitoring
System Owner,Bearing Monitoring System,0.1,4A77E289-F88A-49E0-B1D3-
F3C03B4C589B,Class 350 vehicle 50401 End 1 Bogie wheelset 1 side A,50401-
1-1A,TRUE,771111111,OK,0,1,0,0,0,0,0,0,audio/mpeg3,http://nrdata-
host.co.uk/bearing/fhkd8342k9d.wav,2971,0,Bearing Prob-
lem,8459,Medium,459736,2014-09-22T22:04:21.901Z
2014-09-23T09:49:26.485Z,CSV,restricted,restricted,Bearing Monitoring
System Owner,Bearing Monitoring System,0.1,4A77E289-F88A-49E0-B1D3-
F3C03B4C589B,Class 350 vehicle 50401 End 1 Bogie wheelset 1 side A,50401-
1-1A,FALSE,771111111,OK,0,0,0,0,0,0,0,0,audio/mpeg3,http://nrdata-
host.co.uk/bearing/fhkd876dg57.wav,,,,,,459737,2014-09-23T06:38:58.121Z
2014-09-23T09:49:26.485Z,CSV,restricted,restricted,Bearing Monitoring
System Owner,Bearing Monitoring System,0.1,B4E5E930-2EB0-4D53-B18B-
2B2A8730B8A1,Class 350 vehicle 50401 End 2 Bogie wheelset 2 side A,50401-
2-2A,TRUE,771111221,OK,0,1,1,0,0,0,0,1,1,audio/mpeg3,http://nrdata-
host.co.uk/bearing/fhkd8762bt7.wav,2971,0,Bearing Prob-
lem,8459,Medium,459821,2014-09-22T06:41:26.101Z
2014-09-23T09:49:26.485Z,CSV,restricted,restricted,Bearing Monitoring
System Owner,Bearing Monitoring System,0.1,B4E5E930-2EB0-4D53-B18B-
2B2A8730B8A1,Class 350 vehicle 50401 End 2 Bogie wheelset 2 side A,50401-
2-2A,TRUE,771111221,OK,0,1,1,0,0,0,0,1,1,audio/mpeg3,http://nrdata-
host.co.uk/bearing/fhkd8342k9d.wav,2971,0,Bearing Prob-
lem,8459,Medium,459822,2014-09-22T22:04:23.611Z
2014-09-23T09:49:26.485Z,CSV,restricted,restricted,Bearing Monitoring
System Owner,Bearing Monitoring System,0.1,B4E5E930-2EB0-4D53-B18B-
2B2A8730B8A1,Class 350 vehicle 50401 End 2 Bogie wheelset 2 side A,50401-
2-2A,TRUE,771111221,OK,0,1,1,0,0,0,0,1,1,audio/mpeg3,http://nrdata-
host.co.uk/bearing/fhkd876dh68.wav,2971,0,Bearing Prob-
lem,8459,Medium,459823,2014-09-23T06:39:00.238Z
```

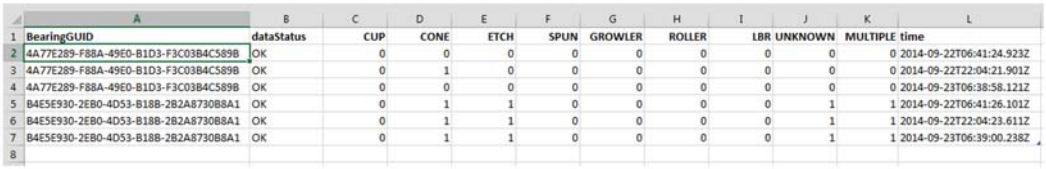
Bearing data response – Concise version

This datagram shows a subset of the data from the previous one, indicating how little needs to be present for the datagram to remain compliant with the data architecture. To be compliant, the mandatory "header" information for the datagram needs to be present: in this case, this would be included in the csv file name, such as "`bearing_monitoring_system_DM_most_recent_2014-09-23T09:49:26.485Z.csv`".

```
BearingGUID,dataStatus,CUP,CONE,ETCH,SPUN,GROWLER,ROLLER,LBR,UNKNOWN,MUL-
TIPLE,time
4A77E289-F88A-49E0-B1D3-F3C03B4C589B,OK,0,0,0,0,0,0,0,0,2014-09-
22T06:41:24.923Z
4A77E289-F88A-49E0-B1D3-F3C03B4C589B,OK,0,1,0,0,0,0,0,0,2014-09-
22T22:04:21.901Z
4A77E289-F88A-49E0-B1D3-F3C03B4C589B,OK,0,0,0,0,0,0,0,0,2014-09-
23T06:38:58.121Z
B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1,OK,0,1,1,0,0,0,0,1,1,2014-09-
22T06:41:26.101Z
B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1,OK,0,1,1,0,0,0,0,1,1,2014-09-
22T22:04:23.611Z
B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1,OK,0,1,1,0,0,0,0,1,1,2014-09-
23T06:39:00.238Z
```

The same datagram would appear in an MS-Excel spreadsheet like Figure 32 .CSV Datagram in MS-Excel.

Figure 32 - CSV datagram in MS-Excel



	A	B	C	D	E	F	G	H	I	J	K	L
1	BearingGUID	dataStatus	CUP	CONE	ETCH	SPUN	GROWLER	ROLLER	LBR	UNKNOWN	MULTIPLE	time
2	4A77E289-F88A-49E0-B1D3-F3C03B4C589B	OK	0	0	0	0	0	0	0	0	0	2014-09-22T06:41:24.923Z
3	4A77E289-F88A-49E0-B1D3-F3C03B4C589B	OK	0	1	0	0	0	0	0	0	0	2014-09-22T22:04:21.901Z
4	4A77E289-F88A-49E0-B1D3-F3C03B4C589B	OK	0	0	0	0	0	0	0	0	0	2014-09-23T06:38:58.121Z
5	B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1	OK	0	1	1	0	0	0	0	1	1	2014-09-22T06:41:26.101Z
6	B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1	OK	0	1	1	0	0	0	0	1	1	2014-09-22T22:04:23.611Z
7	B4E5E930-2EB0-4D53-B18B-2B2A8730B8A1	OK	0	1	1	0	0	0	0	1	1	2014-09-23T06:39:00.238Z
8												

Appendix 3 – References

[1] IETF, “RFC 2104,” [Online]. Available: http://tools.ietf.org/html/rfc2104.

[2] IANA, “MIME Internet Media Types,” [Online]. Available: http://www.iana.org/assignments/media-types/media-types.xhtml. [Accessed 21 05 2014].

[3] IETF, “RFC 5246 Transport Layer Security v1.2,” 2008. [Online]. Available: http://tools.ietf.org/html/rfc5246.

[4] IETF, “RFC 2119 Key words for use in RFCs to Indicate Requirement Levels,” March 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2119.txt. [Accessed 10 06 2014].

[5] IETF, “RFC 6919 Further Key Words for Use in RFCs to Indicate Requirement Levels,” 01 04 2013. [Online]. Available: https://tools.ietf.org/html/rfc6919. [Accessed 10 06 2014].

[6] T1010-01 I, “T1010-01 I Review of relevant RCM developments,” 2014.

[7] R. Fielding, “PhD Dissertation "Architectural Styles and the Design of Network-based Software architectures",” 2000. [Online]. Available: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.

[8] W3C, “PROV Series of Documents,” 2013. [Online]. Available: http://www.w3.org/TR/prov-overview/.

[9] DCMI, “Dublin Core Metadata Element Set, Version 1.1,” 2013. [Online]. Available: http://dublincore.org/documents/dces/.

[10] W3C, “Dublin Core to PROV Mapping,” 2013. [Online]. Available: http://www.w3.org/TR/prov-dc/.

[11] RSSB, “Cyber security in technical systems,” [Online]. Available: http://www.rssb.co.uk/improving-industry-performance/cyber-security.

[12] W3C, “XML Signature Syntax and Processing Version 1.1,” 2013. [Online]. Available: http://www.w3.org/TR/xmldsig-core1/.

[13] W3C, “XML Encryption Syntax and Processing Version 1.1,” 2013. [Online]. Available: http://www.w3.org/TR/xmlenc-core1/.

[14] MIMOSA, “MIMOSA OSA-CBM,” 2006. [Online]. Available: http://www.mimosa.org/?q=resources/specs/osa-cbm-330.

[15] MIMOSA, “MIMOSA OSA-EIA,” 2012. [Online].

[16] IETF, “RFC 4122 A UUID URN Namespace,” 07 2005. [Online]. Available: http://tools.ietf.org/html/rfc4122. [Accessed 28 05 2014].

[17] MIMOSA, “MIMOSA Structured Digital Asset Interoperability Register,” [Online]. Available: http://www.mimosa.org/?q=wiki/structured-digital-asset-interoperability-registry-sdair-functional-requirements.

[18] UIC, “RailTopoModel RC2,” July 2014. [Online]. Available: http://documents.railml.org/science/280714_uic_railtopomodel_rc2.pdf. [Accessed 08 09 2014].

[19] T. Berners-Lee, “The Semantic Web,” 2000. [Online]. Available: http://www.w3.org/2000/Talks/1206-xml2k-tbl/Overview.html.

[20] “FOAF Vocabulary Specification 0.99,” 2014. [Online]. Available: http://xmlns.com/foaf/spec/.

[21] QUDT, “QUDT - Quantities, Units, Dimensions and Data Types Ontologies,” [Online]. Available: http://qudt.org/.

[22] NIST, “NIST SI Units,” 2008. [Online]. Available: http://physics.nist.gov/Pubs/SP330/sp330.pdf.

[23] ISO, ISO 8601:2004 Data elements and interchange formats, ISO, 2004.

[24] W3C, “XML Schema 1.1,” 5 April 2012. [Online]. Available: http://www.w3.org/XML/Schema. [Accessed 18 July 2014].

[25] G. Hohpe and B. Woolf, Enterprise Integration Patterns, Boston, MA USA: Addison Wesley, 2004.

[26] T. Berners-Lee, “Cool URIs Don't Change,” [Online]. Available: http://www.w3.org/Provider/Style/URI.html.

[27] RSSB, “T1010-02 Commercial Framework for Cross-Industry Remote Condition Monitoring Data Sharing,” RSSB, 2014. [Online].

[28] Wikipedia, “UUID,” [Online]. Available: http://en.wikipedia.org/wiki/UUID#Random%5FUUID%5Fprobability%5Fof%5Fduplicates. [Accessed 10 06 2014].

[29] RSSB, “T1010-01 III architecture Requirements,” 2014. [Online].

[30] EPSG, “EPSG Geodetic Parameter Dataset,” [Online]. Available: http://www.epsg-registry.org/. [Accessed 11 09 2014].

[31] UIC, Train Track Interaction Sector, “Harmonization - Running Behaviour and Noise on Measurement Sites,” 2015.

RSSB R&D Programme
Block 2 Angel Square
1 Torrens Street
London
EC1V 1NY

enquirydesk@rssb.co.uk

http://www.rssb.co.uk/research-development-innovation/research-and-development