A Transmodel based XML schema for the Google Transit Feed Specification - With a GTFS / Transmodel comparison

2008.12.29

Nick Kizoom - Kizoom
nick.knowles@kizoom.com

Peter Miller ITO World
Peter.miller@itoworld.com

Control sheet

Document Versions

Date	Author	Version	Description of changes
2008/ 05/02	Nick Knowles	0.1	First Draft from earlier paper
2008/10/22	Nick Knowles	0.3	Revised draft
2008/10/29	Peter Miller	0.4	Revised draft
2008/10/30	Nick Knowles	0.5	Revised draft
2008/11/03	Nick Knowles	0.6	Revised draft with PM & KB initial Comments
2008/12/18	Nick Knowles	0.7	Minor cosmetic changes
2008/12/24	Nick Knowles	0.8	Corrections to Services after midnight from JH
2008/12/29	Nick Knowles	0.0	Further clarification & corrections inc from JH

Related Documents

Ref	Date	From	Version	Document
[Transmodel]	2001/06/01	CEN WG3 TC278	5.1	CEN TC278, Reference Data Model For Public Transport, ENV12896 revised, June 2001.
[IFOPT]	2008	CEN WG3 TC278	1.0	Identification of fixed objects in public transport TC 278 WI 00278207 CEN/TC 278. 2008
[TXC]	Schema guide v43. 17/10/2008 Schema v2.1 2005	UK Department for Transport	2.1 (v43)	TransXChange - An XML Standard for the Data Exchange of Bus Schedules and Related Information. www.transxchange.org.uk
[NaPTAN]	2005/10/10	UK Department for Transport	2.1 (v35)	NaPTAN & NPTG Schema User Guide www.naptan.org.uk
GTFS	2008/08/29	Google		Google Transit Data Feed Specification http://code.google.com/transit/spec/transit_feed_specification.html
[GTFSTm2008]	2007/07/18	Nick Knowles Kizoom	1.1	The Google Transit Feed Specification - Capabilities & Limitations A Short Analysis

Document Automation & Copyright Notice

Kizoom Ltd
109-123 Clifton Street, London EC2A 4LD

Telephone: +44 (0)20 7269 9890
Facsimile: +44 (0)20 7269 9891

Company Registration Number : 3745127

Authors:
Nicholas Knowles, Kizoom
Position: CTO
Peter Miller, ITO World Ltd
Position: CEO

Table of Contents

`

1 Introduction

This paper compares the Google Transit Feed Specification (GTFS) with then equivalent elements from Transmodel and then outlines a proposal to create a Transmodel based XML schema for exchanging transit stop and timetable data. The schema would be fully compatible and interoperable with both Google's GTFS and Transmodel based data sets.

The main purport of this paper - a precise mapping of concepts and terminology between GTFS and Transmodel concepts has value in its own right, even without a schema being created. The additional value of using a "straw-man" schema to make the mapping is that it forces one to be specific and concrete about each and every attribute to be considered - Transmodel itself is an abstract model and leaves many details unprescribed - and also to address how data needs to be grouped in order be serialised efficiently.

Both GTFS and the proposed XML schema can be considered as concrete representations of an underlying abstract conceptual model for passenger transit information. By establishing a common underlying model it becomes easier to exchange data, extend the implementation in future, and to share technology components and tools. In particular as GTFS is enhanced to cover further more complex capabilities, Transmodel can be used to guide and validate the design, drawing on many years of industry experience.

A secondary concern of this paper is to flag the very real practical issues that arise in managing and cross checking complex models serialised into a large number of flat CSV files. Although CSV formats are more compact (though if compression technologies such as gzip are used with XML for over-the wire transmission, this saving is much less material), they put a greater burden on all users to interpret and use the interface correctly, and so in effect couple systems more closely. Our proposal does not in anyway preclude the continued use of CSV, but by establishing an alternative, fully compatible XML representation of GTFS, could help with the validation and consistency of data for many users, since XML allows encapsulation of data into versionable, self describing packages. The use of XML would also make it easier to exchange "deltas" of just the data that has changed (since it is easier to attribute variable metadata to individual elements in XML).

The paper is intended as a discussion draft and some further changes are likely to be appropriate. A number of discussion points are noted.

A successful schema might be helpful as a contribution towards an ISO standard for PT models.

1.1 Background

1.1.1 *Google Transit*

Google Transit is one of a number of innovative Google services such as Google Maps, which provide rich search functions using a free-to-user business model funded by Google's location aware advertising.

In summer 2006 the Google Transit (http://www.google.com/transit) site published a new data exchange format, Google Transit Data Feed Specification (GTFS), through which Transport authorities can make their data available to the Google transit site. Since then the specification has been enhanced through several versions. It has gained considerable momentum, with a number of data sets made available in the USA and many other different countries, and also a growing body of tools such as validators. The original focus of Google transit has been metropolitan area transport networks, comprising primarily bus, metro and ferry. Intercity rail is available and bus information is available in a few countries, such as Switzerland, Austria and Japan.

1.1.2 *Transmodel*

Ever since the invention of the railway, Europe has had the world's densest and most complex public transit networks and the use of public transport is central to all European economies. These networks are highly interconnected, requiring interoperation between many diverse regions and stakeholders; and multimodal, with rail, metro, bus, coach, ferry and other transport modes. As a consequence, European countries have invested significantly in systematic information models to underpin the development of transport information systems and the management of distributed data sets of many different types. The main European initiative, undertaken by CEN, the European standardisation body, is Transmodel.

Transmodel has articulated a comprehensive conceptual model for public transport information systems, considering not just the public facing data that is the main focus of GTFS, but also the other back office and operational systems needed to manage, produce and update both reference and real-time data, so that end-to-end electronic systems can be developed. Transmodel has been used to underpin a number of message sets to exchange particular types of data, such as SIRI (for real-time transport data).

The Transmodel standardisation program has been running since the early 1990s, and has been able to benefit from the extent and the diversity and extent of European transport networks. Examples of almost every different mode, network topology, constraint, fare model, operational model can be found in Europe, which has both single and multiagency configurations operated by both the public and private sector organisations. Many different systems have been compared to establish a common set of flexible abstractions, systematically documented as the Transmodel Corpus. .

As such, Transmodel has already encountered and addressed many of the additional requirements that GTFS is encountering piecemeal as it follows a path already well trodden in Europe. This paper indicates a small number of such further requirements. It can also be remarked that GTFS has already addressed one or two several gaps that were previously identified by comparing the first version of GTFS with Transmodel in our earlier paper.

By using a Transmodel representation it will be possible to extend the GTFS model in future to cover the more complex concepts needed for more complex journey planning including rail system with train parts that join or split and complex interchanges with different routes for mobility-impaired or at different times of the day or week. Otherwise as George Santayana said "Those who forget History are doomed to repeat it".

1.1.3 *Terminology for Public Transport*

One of the achievements of Transmodel has been to establish a precise vocabulary for modelling the Public Transport application domain. This is especially important for Public

Transport where the vernacular uses of terms such as "route", "journey", "trip", or "stop", cover a wide range of overlapping concepts. In the past, different implementors of PT applications have chosen the same terms to describe different concepts - and different terms to describe the same concepts –leading to intense confusion when information systems and data need to be compared. This of course is further compounded by slightly different semantic mappings of concepts into different European languages – and even between different English dialects.

It can be further remarked that some of the most familiar and apparently simple concepts such as a 'route', a 'service' or a 'stop' turn out to have complex set of diverse meanings associated with them

Transmodel assigns more precise technical meanings to its chosen terms, typically reserving the use of a particular word for a single concept. Thus, to give just two examples among hundreds;

- Transmodel distinguishes between a TRIP (the journey made by the passenger), and a VEHICLE JOURNEY (the journey made by the vehicle).
- Within VEHICLE JOURNEY Transmodel distinguishes between a VEHICLE JOURNEY (a timetabled journey in a general timetable that will run on a specific time on a DAY TYPE throughout the period of applicability of the schedule, e.g. Mondays); its resolution as a DATED VEHICLE JOURNEY (a specific journey that runs on a specific calendar date); and also possibly as a MONITORED VEHICLE JOURNEY (representing a vehicle journey that is run under real-time tracking system and so for which real–time observations and predictions and other data are available).

Note that in translating terminology here and elsewhere one should be aware in particular of a few "False friends"; that is, the same term being used in GTFS and Transmodel based standards for what is actually a different concept. For example, "Route" is described in the Google documentation as "a group of trips that are displayed to riders as a single service" a usage that corresponds to a Transmodel LINE i.e. merely a loose descriptive identification of a set of journeys following a similar set of service patterns and marketed to the public as a named service (it is quite possible that not all journeys of a line visit the same stops). Transmodel uses the same term ROUTE for a different purpose; to describe the physical path through the network of a transit Vehicle - a usage that is not abstracted as a reusable concept in the same way in GTFS. One should also be aware that just because a term is used with technical precision in one place, it does not mean it will not be used informally with a different sense elsewhere; for example the use of the term 'service' in the definition of a GTFS-Route just cited above is subtly different from the concept of "service" used to group the calendar days and dates that apply to a GTFS-trip; a given GTFS-*route* may require many different GTFS-*services* to specify the different availability conditions that apply to the various GTFS-trips that follow the route (i.e. make up one single "named service").

1.1.4 Other developments

An important recent development is the CEN TC278 WG3 SG9 proposal to create a concrete XML timetable exchange model based on Transmodel. It will be useful to compare this with GTFS as well.

The European Rail Authority also has an interest in establishing a common XML exchange format for exchanging train data. This raises some additional requirements that GTFS cannot currently handle but which Transmodel can.

1.2 Motivation

Reducing the complexity and cost of managing transport data is essential for developing large scale advanced internet based systems - as recognized by the Google in introducing GTFS. Transport information models are relatively complex and require the fusion of data sets from many different sources. Furthermore these data sets are subject to continual change, both short and long term, which must be processed in a timely manner. Thus not only is it important to have well articulated models, but also to understand how they can be modularised to reflect the actual processes needed to maintain the data - not all of which changes at the same rate or comes from the same source. (For example, stop data may originate differently from schedule data, and change for different reasons).

For any concrete implementation it is also important to pay some attention to robustness and efficiency, i.e. how well the approach works when used to support actual data management processes. Gathering, aggregating and exchanging the diverse data elements inevitably involves the repeated exchange of large datasets over distributed computer systems, using different toolsets and different versions of those toolsets. Most European countries originally developed flat file formats for distributed data exchange (for example the UK's Rail-CIF and ATCO-CIF, or the German VDV 452), but as transport data models have become more numerous and more complex, have found them increasing fragile and hard to evolve. This has led to the increased adoption of XML. Examples of XML formats include TransXChange in the UK, Trident/CHOUETTE in France and work on an XML version of VDV452 in Germany.

XML although not perfect, has some significant technical advantages for data exchange, both in offering a rich object model suitable for representing large and complex models (as found in public transport), and in providing a self-describing, versionable representation suitable for loosely coupled distributed implementations. In particular it offers a degree of encapsulation so that related objects can be packaged together as a coherent whole (this is harder with flat file formats). It allows the reuse of component models, giving implementation and documentation savings. It can be used over many different data transports (e.g. FTP, email, http, TCP/IP sockets) both as a bulk file format (in a similar manner to exchange of traditional flat file format data) and as the payload of protocol requests exchanging just groups of elements (something that is harder to do with flat formats)

The ability to support change needs to be built-in at a fine grained level. The management of transit data is inevitably distributed, involving a chain of connected systems which it is impractical to upgrade simultaneously if there is a data format change. It thus becomes crucial to be able to support multiple versions of data exchange formats simultaneously, and to do this in quite complex ways when different data types are combined (for example a timetable standard may be built on a stop data standard). XML has well supported mechanisms for doing this, making it possible to build reflective systems which can handle multiple version levels.

With agreed data, modelling and exchange standards in place within Europe there is now an emphasis on translating legacy data from proprietary and local formats into standardised formats and on collecting an additional level of detailed data with a confidence that this can be reused and extended even further in future.

1.3 Design Benefits of using Transmodel

One of the main benefits of relating GTFS to Transmodel is that it allows one to use Transmodel as a design tool. That is, it gives one a precise language and conceptual model with which to understand the current capabilities of GTFS. This allows one both to identify limitations with the current model and to suggest future enhancements based on proven representations that correspond to industry practice (and so be able to utilise more data in future).

This section summarises some specific examples - aspects of many of these are also discussed in the analysis of the GTFS model in the next section or noted in Appendix A.

1.3.1 Rail Services with splitting trains

Trains are often made up of train parts that may split or join along a route. Some understanding of this joining and separating is needed in order to plan and describe journeys on such services. It may be necessary for people heading to different destinations to board the same VEHICLE (i.e. train), but possibly in different sections of the train, for example 'the front four carriages'.

Transmodel includes elements such JOURNEY PARTs and COUPLED JOURNEYs that permit the description of these additional linked journeys in the schedule, as well as elements to describe the physical construction of TRAINS into TRAIN ELEMENTS such as carriages This allows journey planning and the production of trip itineraries that correspond to how journeys in the real world.

Using GTFS it is unclear how one should model linked journeys. Should multiple VEHICLE JOURNEYs be created for each of the various possible routes that someone could take without changing seat? How is this described to the user?

1.3.2 Multi-operator services

In Europe it is common for services in one area to be operated by a number of different commercial companies and in some cases two or more bus operators may share out the operation of a single LINE with one running it commercial during peak periods and another running it as a 'tendered service' in off-peak periods.

Using GTFS it is unclear how one should best model multi-operator operation. If one simply assigns them to the same AUTHORITY one loses information about the operator. If one treats each OPERATOR as a separate AUTHORITY one loses information about connectedness (and there may be data fusion issues).

In Transmodel there are distinct OPERATORs and GROUPs of OPERATORs to define the organisations running the services and individual VEHICLE JOURNEYs can be associated with the appropriate OPERATOR. Note that a particular LINE can consist of VEHICLE JOURNEYs that may be operated commercially by one OPERATOR during peak periods and by a different OPERATOR in off-peak periods under contract to the AUTHORITY. A current example of this problem is in the South East region in the UK which operates in this way.

1.3.3 Stop Areas, Connection Links Interchanges

On multi-leg trips it is often necessary to alight from a service at one STOP POINT and board another service from a different STOP POINT. In some cases both STOP POINTs may be close to each other on the same side of the road, in others the user may need to cross a street or walk road a corner which may or may not be easy and obvious, in other cases it might involve a long walk with multiple escalators, not all of which may be accessible to all types of user. The proximity of two STOP POINTS provides no guarantee that there is a link between them, there may be an obstruction such as railway line, stream, wall or busy road or the STOP POINTs may be at different heights with a difficult or impractical route between them.

Transmodel uses a number of different methods to provide information about these issues. A STOP AREA can be defined to associate a number of STOP POINTs that are 'close to one another' and most STOP POINTs will be part of a STOP AREA, often one on the other side of the road. STOP AREAs can be contained other stop areas and around a busy junction individual STOP POINTs may be clustered into a number of STOP AREAs which are then grouped into another STOP AREA. A station may then be part of a hierarchy of STOP AREAs which include other services available nearby at street level. For example, St Pancras station is made up of a domestic and an international area, or most airports are made up from several terminals, a metro station a bus station, etc.

General time-related quality parameters for interchanges for an interchange can be specified using a DEFAULT INTERCHANGE - see also Connection Protection below.

Transmodel also allows specific transfer times between any two STOP POINTs using a CONNECTION LINK which gives the transfer times for various different sorts of user; including frequent uses, occasional users and mobility impaired users. For complex interchanges the IFOPT standard allows more detailed information to be collected about the physical topology, see 'Station and Transit Interchange Navigation and Accessibility' below.

GTFS allows uses to create Stops that are either the equivalent of STOP POINTs or Stations, however it is unclear how STOP AREAs that are not actually stations should be encoded and GTFS does not allow more than two levels of hierarchy (stations and stops). Within GTFS there is also a *transfer* element (added in 2008) that allows one to transfer links and provide a minimum transfer time. This 'transfer' element is also used to provide information about a separate concept, that of a protected connection where the departing vehicle may wait for the arrival of a late incoming service. Journey connections are treated as a distinct concept by Transmodel (see 'Protected Connections' below for details).

1.3.4 Station &Transit Interchange Navigation and Accessibility

Large stations, airports and other interchanges have complex internal topologies that are subject to complex constraints as to accessibility and availability. For example an entrance may only be open at certain times, or be accessible to certain types of user, or a subway may have one way foot tunnels that become congested at certain times of day. Long platforms may have specific boarding points, the location and availability of parking may also be relevant.

Advanced journey planning needs to take into account such accessibility. A recent subproject of Transmodel, IFOPT, has been to devise a model for representing such aspects of interchanges. Collecting such data (which typically requires an internal survey by the transit operator) will take some years so it is important the models are still useful when partially populated.

Bank and Monument tube station in London is a good example of the need to model interchange information in some detail. In this case as it is in many respects two separate stations connected by very long pedestrian tunnels. It has a number of entrances each of them with a cluster of bus stops.

Significant parts of Europe's rail and metro systems are over 100 years old and were built before any consideration was give to the access needs of mobility-impaired wheel-chair users. For example, many of London's underground stations are not accessible to wheel-chair users although newer stations are now accessible. On many rural stations, a wheel-chair user will not be able to access some platforms as there is no lift, and on others will need to allow additional time to be escorted across the tracks.

GTFS allows for a simple 'station' and 'stops' with no method of representing this more detailed information some of which is essential for mobility impaired users.

1.3.5 Connection Protection or Guaranteed Connections

Within public transport systems that link up journey networks, certain services may be intended to connect, and even delay their departure at certain stops to wait for a delayed incoming service. It might be that that a bus to local villages will delay by up to 5 minutes for a delayed main line train, or that the last bus from the station will wait up to 30 minutes for the incoming service.

It appears that the GTFS-*transfer* element in effect conflates this concept (of particular journeys connecting) with the related but distinct concept of a CONNECTION LINK (i.e. of it being physically possible to get between two stops in a certain time, that is, all journeys connecting).

Within Transmodel, the term 'interchange' is used to describe the recognised/organised opportunity for people to change between different vehicle journeys at the same or nearby STOP POINTs. A SERVICE JOURNEY PATTERN INTERCHANGE is used for define the connection between all vehicles operating on two different JOURNEY PATTERNs and the SERVICE JOURNEY INTERCHANGE is used to define the connection between two individual VEHICLE JOURNEYs. It can also be used to indicate whether journeys officially connect, with distinctions being possible between Planned, Advertised and Guaranteed connections, where *planned* indicates it is the intention that passengers can make the connection, *advertised* that specific claims are made in public about the possibility of a transfer and *guaranteed* that the second vehicle will wait for a defined period for a delayed incoming service. The SIRI standard defines a 'Connection Timetable' service to exchange information about planned connections and a 'Connection Monitoring' service to monitor connections in real time.

In the GTFS-*transfer* file GTFS allows for a guaranteed connection to be created on a particular CONNECTION LINK for a GTFS-type 1 transfer, however in most circumstances only certain VEHICLE JOURNEYS will wait for certain incoming services and it is not possible to express this within a entity that is more similar to a CONNECTION LINK than to a SERVICE JOURNEY INTERCHANGE or SERVICE JOURNEY PATTERN INTERCHANGE (see above in the 'Interchanges' section.

1.3.6 Services across Midnight

Busy and vital urban areas often have services that run into the early hours of the morning or even round the clock. There are a number of subtle considerations as to how to best to represent what happens over midnight and how transit companies handle the hand-over between weekdays services and weekend services, or Sunday evening services to the Monday service.

In GTFS, individual VEHICLE JOURNEYs are allowed to continue beyond midnight (and indeed a VEHICLE JOURNEY may take more than 24 hours), and indeed to start beyond midnight by using the artifice of a "clock value" greater than 24 hours for example 27:00 or 28:20 to indicate they are still part of the previous day's operations.

Transmodel uses a more explicit abstraction to capture this concept - the OPERATING DAY. In Transmodel, an explicit OPERATING DAY can define a reusable day with a start and end time that may not correspond to midnight and typical 3am to 3am may be used. A specific journey may thus be associated with a particular OPERATING DAY. In areas where 24 hour operation is provided then the OPERATING DAYs of two subsequent dates may overlap, allowing the services on a particular Friday night to not complete until 3am on the following morning, but allowing the first of the Saturday services to start prior to midnight on Friday. The resolution of an OPERATING DAY may be associated with a DAY TYPE; on a DATED VEHICLE JOURNEY this is resolved into a specific calendar date in a specific year.

In the GTFS the concepts of DAY TYPE and OPERATING DAY are in effect conflated on GTFS-*calendar* with the temporal boundaries of the "OPERATING DAY" being derived from the individual journeys associated with the corresponding GTFS-*service*. This leaves some information relevant for presenting and combining journeys unexpressed.

1.3.7 Services across Time-zones

Some rail, ferry and coach services operate across time-zones. Examples are ferries between the UK and the continent, and Eurostar rail services between London and Paris.

In GTFS the times for individual calling points are given relative to the time zone specified by the agency. A time-zone is defined within the GTFS-*agency*; however there is an assumption that the Agency is only operating is a single time zone and the method used to give the calling times at different stops in different time zones is not clear (though there is a proposal pending to address this).

Time-zones really need to be at a stop level, and to avoid mistakes in handling journeys that cross time-zones, or journeys that run across a clock change, UTC should always be used.

1.3.8 Day Types

The GTFS-*service* allows one to specify the availability of a VEHICLE JOURNEY as running on particular days of the week (specified by a *GTFS-calendar*) with exceptions on specific dates (specified by *GTFS-calendar_dates)*. Transmodel uses a slightly more general notion of a DAY TYPE which may be a DAY of WEEK (as for GTFS) or may be some other type of day that affects demand for services, for example "Market day", or "Match Day". This allows important additional availability conditions to be captured.

GTFS-calendar and *GTFS-calendar_dates* assume a single set of DAY TYPEs. Transmodel allows arbitrary additional ones to be added.

1.3.9 Stop Labelling

Most real-live public transport systems have found it valuable to be able to use a number of different name styles for a given for STOP POINT in different circumstances. The name may need to be rendered differently in different contexts, either to discriminate it from other stops, or to accommodate different display devices.

Experience in the UK with the NaPTAN standard has shown the importance of allowing names to be constructed from a number of different elements. By way of example, a bus stop at the railway station in Cambridge in Cambridgeshire (there is another Cambridge in Gloucestershire) has a full name of "Cambridge (Cambs), Railway Station (Stop A)", however in the context of someone already in the city, the appropriate description would be "Railway Station (Stop A)". For journey planning purposes in the context of the east of England the name "Railway Station, Cambridge" may be sufficient. In the context of a rail journey planner, "Cambridge" alone might suffice. The European IFOPT standard similarly has a number of name elements and classifiers (such as the transport mode) and the bearing (direction of the road by the stop) that are useful for describing it.

The labelling of stops supported by GTFS is quite basic with a single field for 'name' and another for a short 'code'. One cannot even determine the transport mode of the stop without recourse to a vehicle journey that uses the stop. Although the name can be populated in the GTFS format by combining the various discrete elements into a single long string, doing so loses information about the discrete parts of the name that would allow a better rendering in some applications.

1.3.10 Transport Modes, Vehicle Equipment & Accessibility

Public transport in Europe, as elsewhere can be divided into different modes, for example bus, train and tram. A particular mode there can be further distinctions, for example double decked bus, low floor bus and minibus within the transport mode bus. In addition optional equipment may or may not be available on a particular vehicle, for example audio announcements or a wheel-chair loading ramp. Bus Stops are also increasingly being converted to allow no-step wheel-chair access to appropriate buses.

Laws such as the UK's Disability Discrimination Act require that organisations make services accessible to disabled users, and in some areas local authorities are paying for new low-floor accessible buses to run on particular routes.

Transmodel uses TRANSPORT MODE to define the main mode (bus, ferry etc). Transmodel also allows particular VEHICLE TYPEs to be assigned to different SERVICE JOURNEYs and for different TYPES OF EQUIPMENT to be fitted to different VEHICLE TYPEs.

GTFS uses 'Route type' is equivalent to TRANSPORT MODE but misses some significant European modes. It doesn't distinguish between bus and coach, or between long distance and suburban rail so using Google Transit in the UK services run by National Express are described as 'bus'. These distinctions are important to users in understanding the nature of the service, the likely cost and relation to any travel passes they may have, etc. GTFS has no mechanism to exchanging information about vehicle types or vehicle equipment that are relevant to mobility impaired travellers including those in wheel-chairs and those with young children in push-chairs.

1.4 References & Examples

1.4.1 GTFS

The Google Transit Feed specification can be found at

http://code.google.com/transit/spec/transit_feed_specification.htm

This document is based on Google Transit Feed Specification as of August 2008.

1.4.2 Transmodel

The CEN standard Transmodel specification can be found at http://www.Transmodel.org/
The Identity of Fixed Objects in Public Transport IFOPT is a new CEN Technical Standard work item that extends refines some Transmodel concepts for Stop Places, that is stations, stops and transport interchanges http://www.naptan.org.uk/ifopt/

1.4.3 Previous work

This paper is a development of a comparison originally made as a short paper in 2006 and updated in 2007 "The Google Transit Feed Specification - Capabilities & Limitations a Short Analysis" [TM/GTS2007] based on the 2007 version of GTFS. This mainly considered GTFS against Transxchange (for bus timetables)

There already exist tools to convert between GTFS TransXChange (a Transmodel based schema used in the UK) - these provide a ready existence proof of Transmodel & GTFS interoperability.

1.5 Presentation conventions

We follow conventional practice of using UPPER CASE for Transmodel entities and ***BoldItalicCamelCase*** for XML Schema Elements from IFOPT, SIRI, etc. We use GTFS-*italic* as prefix where we want to indicate a GTFS entity e.g. GTFS-*route*

1.6 Acknowledgements

The paper has been prepared by Nick Knowles, CTO of Kizoom. I am grateful to Peter Miller of Ito World Ltd who sent me his own independent notes while I was working on both the original and the revised papers, and who has contributed to this second paper, and to Kasia Bourée, the leading French Transmodel expert, for her very helpful comments and corrections and Roger Slevin for his guidance and own expert review comments. I thank also Joe Hughes of Google for his important corrections and clarifications on interpreting GTFS.

2 A Transmodel GTFS model

2.1 Approach

This goal of this paper is to (i) establish a mapping between GTFS and Transmodel terms; (ii) to express the contents of the GTFS model in a minimal subset of Transmodel; and (iii) to suggest an XML schema structure with which GTFS could be encoded.

We do this as follows

(i) Establish a mapping between the current GTFS and Transmodel

- Outline the contents of current GTFS.
- Present a simple "one page" UML model of the current GTFS entities with both GTFS and TM terms indicated.
- Annotate the GTFS specification in detail with Transmodel terms (See Annex A).

(ii) Express GTFS as a revised Transmodel based model

- Present a simple "one page" UML: model of the current GTFS expressed using Transmodel entities. This involves renaming some elements and breaking down a couple more into Transmodel abstractions.

(iii) Sketch out a possible Schema structure: this is largely a matter of Noting which are the key elements to expose as root elements in a schema

2.2 Overview of Current GTFS Components

GTFS is for the static exchange of public transport stop and schedule data.

Each version of schedule data is sent as a set of CSV tables, encapsulated as a zip file. Table 2-1 shows the CSV tables making up the GTFS. On the rightmost column are the equivalent Transmodel concepts found in each file.

V1 indicates a file or element that was in the Nov 2006 version of the GTFS. V2 indicates versions added up to June 2007, V3 in the August 2008 version.

	Google File		**Contents**	**CEN Transmodel Concept**
V1	**agency.txt**	Required	Information about the transit agency.	OPERATOR AUTHORITY
V1	**stops.txt**	Required	Information about individual locations where vehicles pick up or drop off passengers.	SCHEDULED STOP POINT, STOP PLACE (IFOPT), TARIFF ZONE
V1	**routes.txt**	Required	Information about a transit organization's routes. A route is a group of trips that are displayed to the rider as a single "service".	LINE
V1	**trips.txt**	Required	Information about scheduled service along a particular route. Trips consist of two or more stops that are made at regularly scheduled intervals.	VEHICLE JOURNEY
V1	**stop_times.txt**	Required	Lists the times that a vehicle arrives at and departs from individual stops for each trip along a route.	Call PASSINGTIMES STOP POINT IN JOURNEY PATTERN (SERVICE PATTERN) (ROUTE LINK - distance)
V1	**calendar.txt**	Required	Defines service categories. Each category indicates the days that service starts and ends as well as the days that service is available.	DAY TYPE PERIOD, DAY OF WEEK
V1	**calendar_dates.txt**	Optional	Lists exceptions for the service categories defined in the **calendar.txt** file.	OPERATING DAY
V1	**fare_attributes.txt**	Optional	Defines fare information for a transit organization's routes.	FARE ELEMENT PRICE
V1	**fare_rules.txt**	Optional	defines the rules for applying fare information for a transit organization's	FARE ELEMENT, DISTANCE MATRIX
V2	**shapes.txt**	Optional	Provides rules for drawing lines on a map to represent a transit organization's routes.	ROUTE ROUTE LINK / PROJECTION
V2	**frequencies.txt**	Optional	Provides the headway (time between trips) for routes with variable frequency of service.	(Frequency)
V3	**transfers**	Optional	.Provides additional rules for making connections between routes.	CONNECTION LINK, SERVICE JOURNEY INTERCHANGE, SERVICE JOURNEY PATTERN INTERCHANGE DEFAULT INTERCHANGE

Table 2-1 Google Transit Feed Specification tables

2.2.1 *Overview of GTFS Model*

The GTFS does not have a formally describe model of entities and their relationships, but an implicit model can be inferred.

Figure 2-1 shows the current GTFS model "reverse engineered" from the GTFS specification. In the diagram, the GTFS-file name is used to name each element, with equivalent Transmodel terms shown in brackets in upper case on some of the entities. Colours are used to indicate the main content areas (Green = Stop data, Yellow = Schedules, Blue= Fares, grey = base data types, etc). Note that some entities (e.g. GTFS-*service*, GTFS-*block*) are referenced in GTFS but not otherwise described by a GTFS file.

- A GTFS-*agency* (i.e. AUTHORITY) operates one or more named GTFS-*routes* (i.e. Transmodel LINEs).
- GTFS-*trips* represent individual VEHICLE JOURNEYs that follow a LINE.
- The GTFS-*trips* can be grouped with a GTFS-*service*. This is an arbitrary group of journeys that share a common VALIDITY CONDITION.
- GTFS allows the validity condition to be specified both as GTFS-*calendar* (i.e. DAY TYPEs) and GTFS-*calendar_dates*, i.e. specific OPERATING DAYs the service does or does not operate.
- Each trip is made up of a number of GTFS-*stop-times* comprising times at GTFS-*stops*. These correspond to CALLs (i.e. TIMETABLED PASSING TIMES for STOPs IN JOURNEY PATTERN) at a SCHEDULED STOP POINT/ STOP PLACE ELEMENT.
- Each GTFS-*stop* can be assigned to a GTFS-*zone* (i.e. TARIFF ZONE). GTFS-*fares* can be specified for zone to zone journeys. This corresponds to a FARE ELEMENT associated with of a DISTANCE MATRIX.

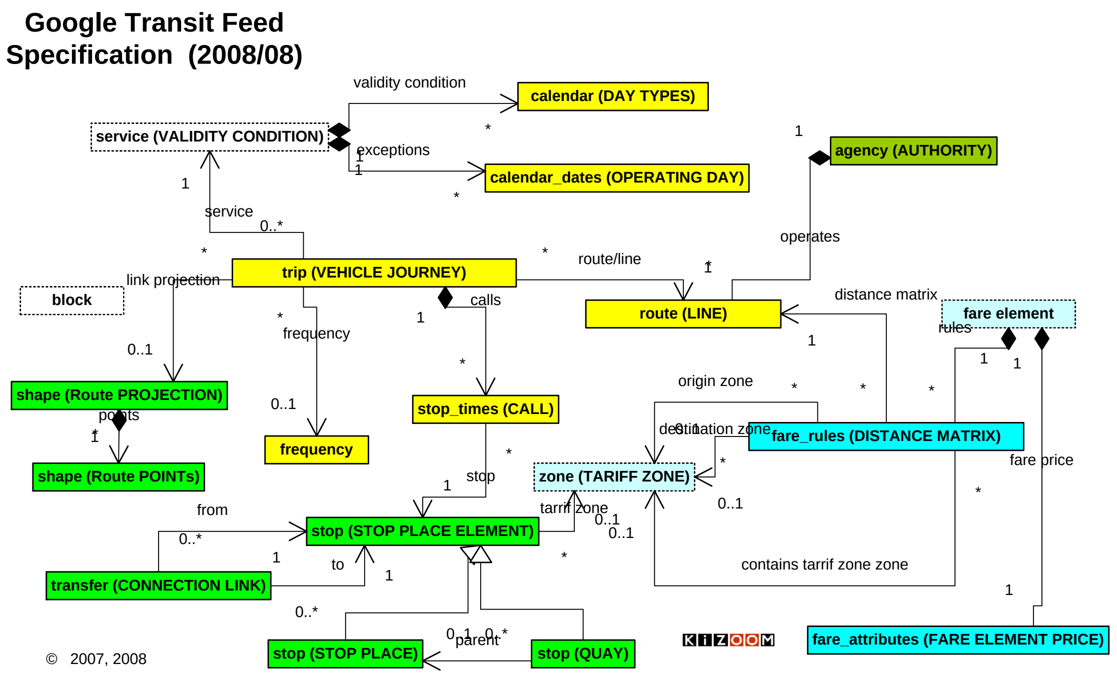

Figure 2-1 Summary of GTFS model

2.2.1.1 Notes on terminology Vis a Vis Transmodel

- **GTFS-*trip***. GTFS uses the term 'Trip' for the journey made by the transit vehicle. Transmodel reserves TRIP for the journey made by the passenger (taking one or more VEHICLE JOURNEYs), and uses VEHICLE JOURNEY for the journey made by the transit VEHICLE (Be it Train, bus or ferry). This avoids confusion when considering models for the results produced by journey planners.

- **GTFS-*service*** is used in GTF as a mechanism to group an arbitrary set of journeys that have the same temporal constraints such as period and days of operation. This may or may not correspond to a 'service' in the colloquial sense of a "timetable" of journeys for a particular route. Quite often a real-world "Service timetable" will comprise a number of journeys with the same temporal constraints together with a few with exceptions or differences shown as footnotes (e.g. "not on Wednesdays"). In GTFS these would be two separate GTFS-*services*. Because 'Service' has many connotations of wider common properties in colloquial English it would be better to use a different term. Thus if the GTFS-*service* indicates only a temporal constraint (as appears to be the case), then it would be better to map it to the more specific Transmodel VALIDITY CONDITION. If a wider sense is needed then GROUP OF JOURNEYs might be an alternative. If it is the entire set of journeys, then TIMETABLE VERSION would be the appropriate element.
  - ➔ DISCUSSION POINT: *Confirm we should use VALIDITY CONDITION and not say GROUP OF JOURNEYS?*
- **GTFS-*route***. GTFS uses the term 'Route' for the arbitrary public label given to a set of journeys. This appears however to correspond more closely to Transmodel LINE. (LINE: A group of ROUTEs which is generally known to the public by a similar name or number.) In Transmodel, a ROUTE is an ordered list of located POINTs defining one single path through the road (or rail) network. A ROUTE may pass through the same POINT more than once. It is an abstract concept, different from that of the physical path. Thus Transmodel uses the term ROUTE for the specific set of points and route links that constrain a JOURNEY PATTERN & VEHICLE JOURNEY Note in particular that ROUTE is directional. For example: two ROUTEs defining the same path but with opposite directions will often belong to the same LINE.
  - ➔ DISCUSSION POINT. *Confirm we should use LINE, not ROUTE?*
- The **GTFS-*route_type***. This appears to be a Transmodel MODE of transport.
- The **GTFS-*shape*** is specific to the individual VEHICLE JOURNEY (<u>and not the overall ROUTE</u> which may be covered by many different VEHICLE JOURNEYs using different JOURNEY PATTERNs) and provides a plot of points. However it also includes link related data - GTFS-*shape-distance-travelled* - that is common to all JOURNEY PATTERNs that follow the route (not just this VEHICLE JOURNEY), so it might be considered to be a view of a set of ROUTE POINTs and ROUTE_LINKs.
- **GTFS-*zone*** is in effect a TARIFF ZONE. The GTFS-*fares_rules* describes a DISTANCE MATRIX of zone to zone transitions, in Transmodel these are types of FARE STRUCTURE ELEMENTs.
  - ➔ DISCUSSION POINT: DISTANCE MATRIX *or* FARE STRUCTURE ELEMENTs*?*
- **GTFS-*fare_attributes*** describes a FARE ELEMENT PRICE which may be associated through a FARE ELEMENT with a DISTANCE MATRIX element.

2.2.1.2 Notes on model

The GTFS provides a basic timetable representation. Transmodel considers a number of more elaborate capabilities. For example, among the current limitations of GTFS are the following

- A constraint or preference on transferring can only be placed on all movement between two GTFS-stops, not on an individual trips/VEHICLE JOURNEYs, i.e. GTFS does not make the Transmodel distinction between the CONNECTION LINK and the SERVICE JOURNEY INTERCHANGE which may take place over the CONNECTION LINK between specific VEHICLE JOURNEYs.
- Similarly, either all or no transfers between journeys at a stop can be GUARANTEED. Transmodel distinguishes between PLANNED, ADVERTISED and GUARANTEED (i.e. "connection protection") transfers.

- GTFS does not have mechanism to describe train journeys that break or join for part of their journey, as per Transmodel JOURNEY PART, COUPLED JOURNEY, TRAIN PART etc.
- A GTFS-*stop* can only be in a single tariff zone.
- There is no separation of the service pattern or journey pattern from the pattern of stops made by an individual vehicle journey (so no reuse of information layers). GTFS uses a point in sequence rather than link model (which means it cannot support different data that applies to the use of the link rather than the stop). This is fine just for final passenger timetables, but it would be helpful to reference a journey pattern to be able to cross reference back to the underlying timing pattern in some circumstances (e.g. for interpolating times), - just as say, the BLOCK is referenced though the GTFS-*block_id*).
- There is no versioning or other metadata on elements. So a complete data set has to be provided every time and it is not possible to distinguish or compare different data versions.
- All data identifiers are assumed to be unique within a GTFS-*Authority*. This is sufficient to ensure data references are unique within GTFS's engines (assuming Google assigns a unique scope to each Authority and manages this on a global scale.), but does not provide any ready means by which an Authority might reference the resource of another - such as a shared stop, or a create connecting journeys in a timetable. One may note that for some modes there are already supranational agreements for allocating certain identifiers (e.g. IATA airport codes, or UIC station location codes), however for bus and coach, which have many more stops and timetables, this would be impractical. A GTFS-*Agency* has no inherent connection to a political division, so could span a border. (IFOPT addresses this issue by allowing data elements to be associated with an administrative jurisdiction.
- DAY TYPEs other than day of the week are not supported (e.g. market days, match days, public holidays). Although the precise condition of availability on a given calendar date can be specified by the use of GTFS-*Calendar*-dates to specify that a GTFS-*Trip* does or does not run on a specific date, the pattern of use, nor the passenger facing explanation (e.g. "Runs on Market days only") is not captured.
- A GTFS-*stop* is either a point of access (i.e. QUAY) or a station or stop pair (i.e. STOP PLACE).
- OPERATOR (as opposed to AUTHORITY) data is not captured, even if an AUTHORITY runs services provided by multiple OPERATORs.
- Projection of stops onto topographic area is purely via geo-coordinates, and not accompanied by explicit association with topographic locality, as say supported by IFOPT (and as say used in the UK between NaPTAN stops and the UK National Public Transport Gazetteer - NPTG - localities). Google is of course able to use its own location and point of interest model for this purpose. However PT gazetteer models such as those in NPTG and IFOPT formalise additional location concepts that are specific for travel and that may involve more than simple proximity. (For example airports and stations are not necessarily physically located in the places that they serve). Or only certain stations count as "main termini".

2.2.2 Detailed GTFS Model

Figure 2-2 shows a more detailed model of the current GTFS feed, with attributes and enumerations shown.

- GTFS has equivalents to the basic Transmodel attributes such as identifiers, and names as well as the fundamental properties that characterise specific entities for example arrival and depart times, or point coordinates
- GTFS includes a number of useful additional attributes such as urls. In general a GTFS schema need aim only to support the attribute specifically required by GTFS.

- The choice of enumeration values etc is not prescribed by Transmodel. However some of the concrete implementations of Transmodel, such as SIRI and IFOPT have preferred Transmodel encodings.

Appendix A annotates all the elements of the GTFS specification with detailed Transmodel equivalents.

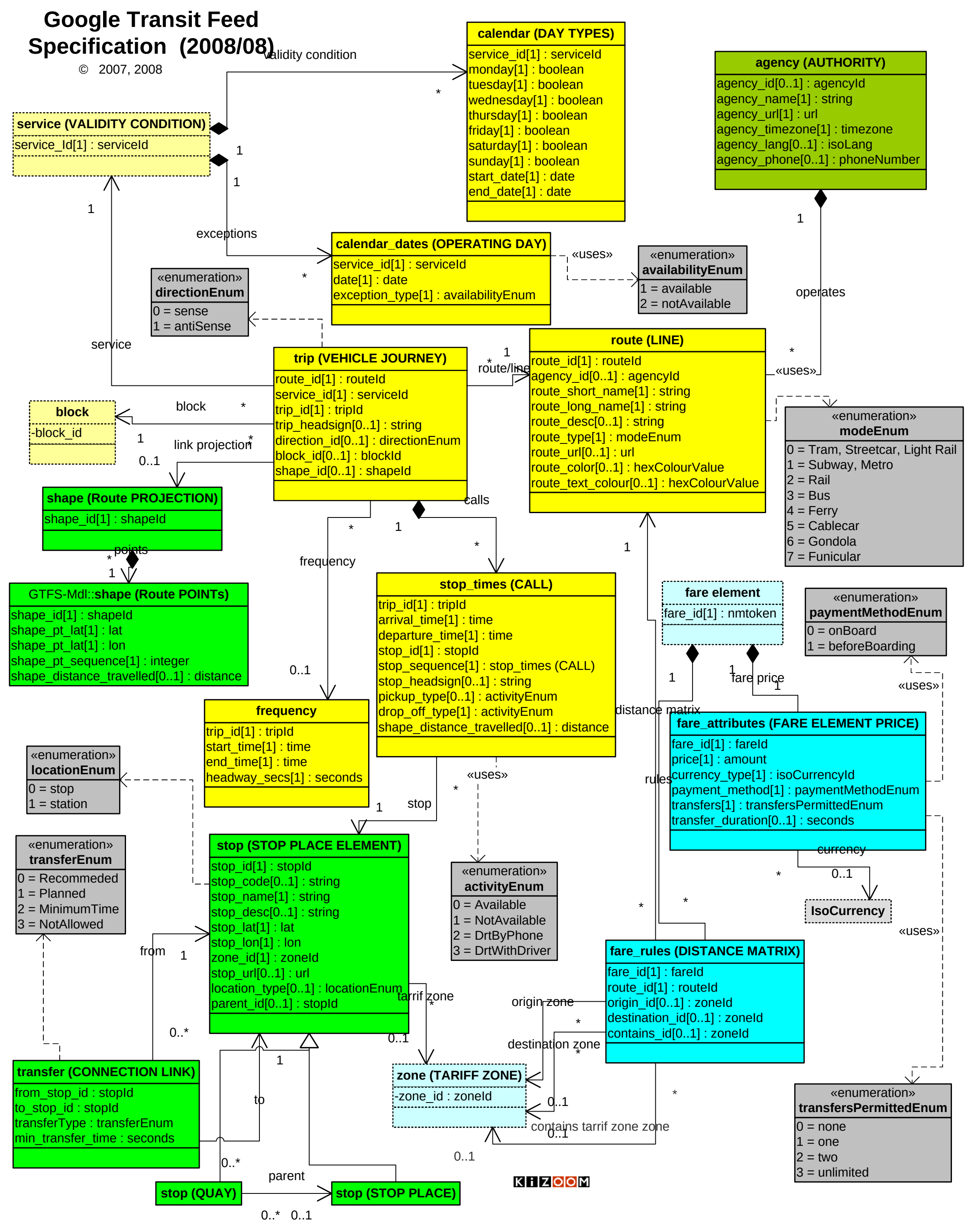

Figure 2-2 Detailed model of GTFS

2.2.2.1 Notes on terminology

- The ***GTFS-route_type*** attribute is a Transmodel mode

2.2.2.2 Notes on attributes

- The GTFS-route_type (i.e. Transmodel MODES) lacks Coach, and Air modes. In Europe the distinction between long distance Rail and light Rail/ Suburban rail (as distinct from tram/ street car) is also important. The use of Cable car for Streetcar pulled by is cable is curious to San Francisco.

2.2.3 Data types

GTFS uses a limited number of data types and enumerations. These are summarised in Figure 2-3.

Identifiers of entities must be unique within their type and agency so we show them as logical types, even though they are arbitrary strings.

Direction has no inherent meaning of outbound/inbound

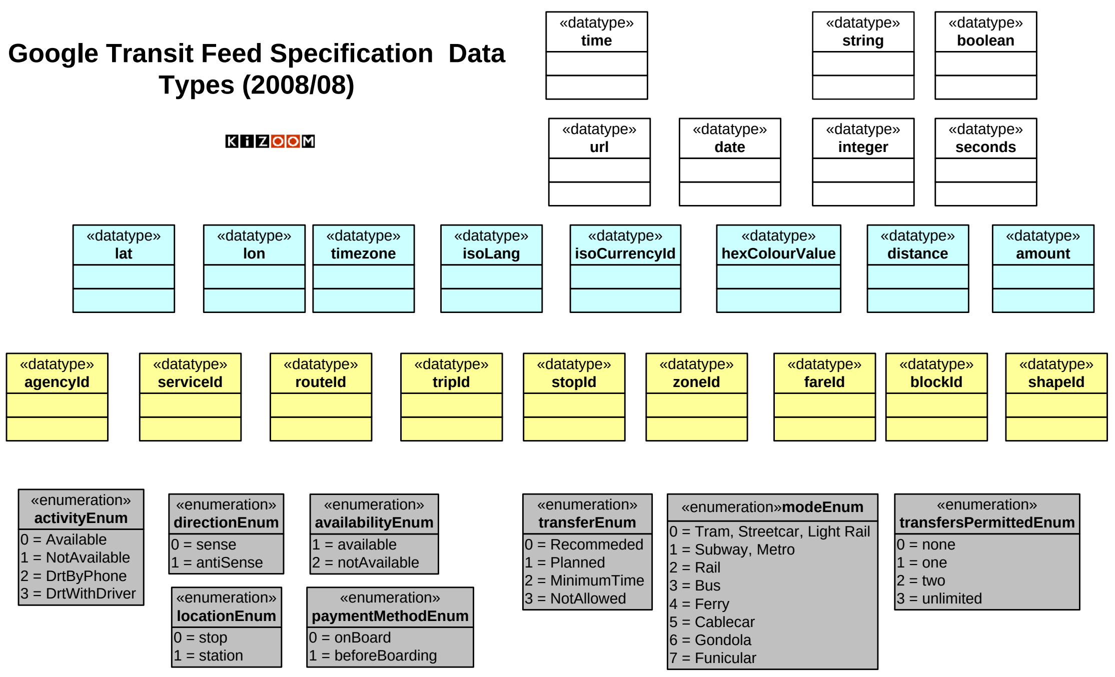

Figure 2-3 GTFS-Data types

2.3 GTFS in Transmodel

Figure 2-4 shows a slightly revised model that expresses GTFS using Transmodel terms. (For reference, GTFS terms are shown in brackets). This can be compared with Figure 2-1.

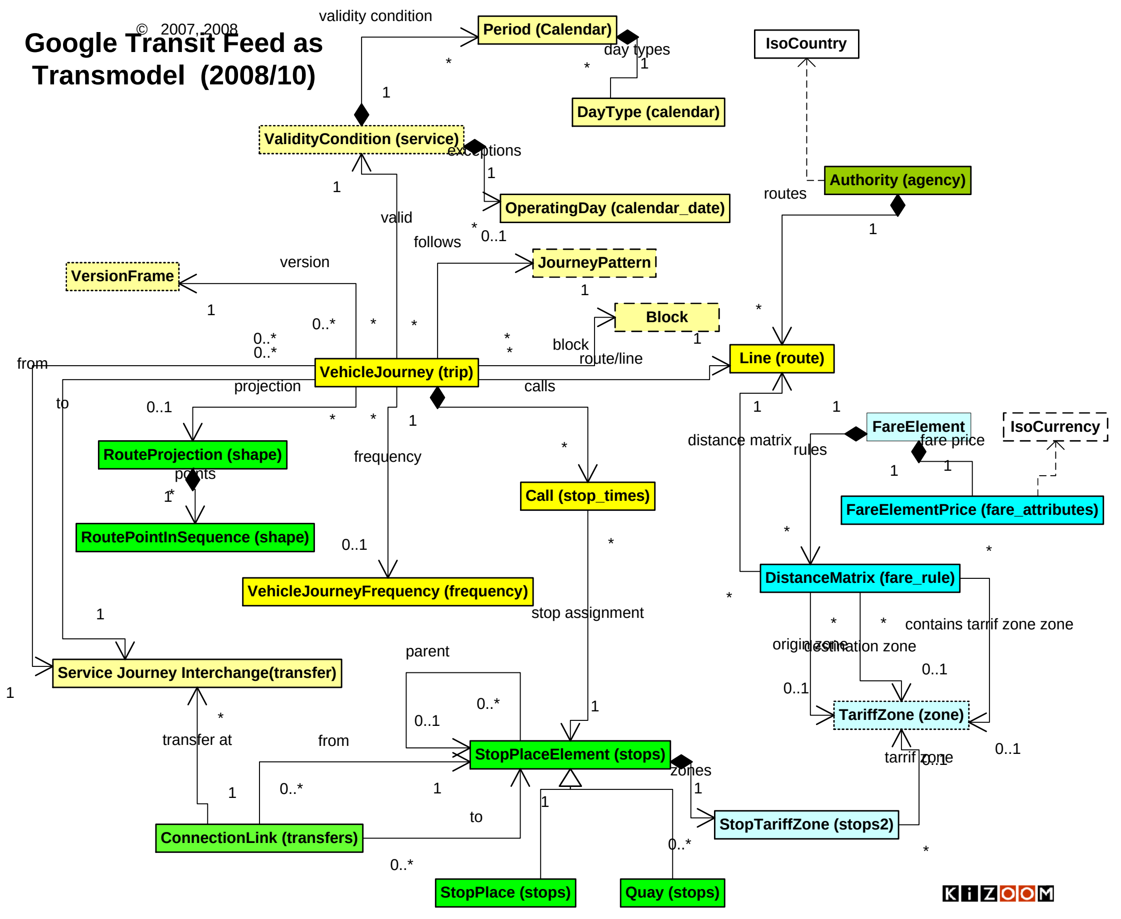

Figure 2-4 GTFS using Transmodel terminology

2.3.1.1 Notes on Model

The proposed "Transmodel GTFS" model is close to that of GTFS, with a number of possible small refinements. Some of the differences are just name changes; others are minor structural refinements giving more generality, which would be backwards compatible.

- GTFS-*trip* is renamed VEHICLE JOURNEY; it contains an ordered sequence of CALLs corresponding to GTFS-stop-times. (A CALL is Transmodel view that assembles PASSING TIMEs and other journey attributes for each stop - GTFS also includes a ROUTE LINK attribute - distance from last stop)
- GTFS-*service* is renamed to be more specifically a VALIDITION CONDITION, i.e. a condition that can be shared by and arbitrary group of journeys with shared availability. (It might be appropriate to call this GROUP OF JOURNEYS instead if there were other properties in a GTFS-service - see earlier discussion).
- The GTFS-*calendar* (which currently only has DAY OF WEEK) is normalised into a PERIOD and DAYTYPE. This allows other DAY TYPEs (e.g. match days, market days, etc to be supported).
- GTFS-*zone_id* is explicitly reified as a renamed TARIFF ZONE. Use of zone is normalized from GTF-*stop* into a separate ***StopTariffZone*** element so that a stop can potentially belong to more than one TARIFF ZONE.

- GTFS-*transfer* is renamed CONNECTION LINK (i.e. the possibility of transfer between two STOP POINTs regardless of vehicle journey) and then as an enhancement, refined into an additional distinct abstraction: SERVICE JOURNEY INTERCHANGE, the possibility of interchange between particular VEHICLE JOURNEYS. This may be GUARANTEED.
- GTFS-*shape* is considered to be a ROUTE PROJECTION (although strictly speaking it is only the shape of an individual VEHICLE JOURNEY, and not the whole ROUTE). For modelling purposes is normalised into a parent and child (GTFS-shape.txt in effect holds just the children) though there are no interesting attributes of the parent.
- GTFS-fare_*id* is explicitly reified as a renamed FARE ELEMENT. A FARE ELEMENT associates a FARE ELEMENT PRICE with (corresponding to GTFS-*fare_attributes*) with a zone to zone DISTANCE MATRIX element. (Possibly this should be named a FARE STRUCTURE ELEMENT(
- We allow nesting of Stop Place elements so that large interchanges can be built up from smaller ones.
- A VEHICLE JOURNEY can reference a JOURNEY PATTERN, allowing it to be tied in to the underlying timetable model.
- GTFS-*agency* is renamed AUTHORITY. The AUTHORITY can indicate in which country it is.

2.3.2 *Detailed Transmodel GTFS*

Figure 2-5 shows detailed attributes for the model from **Figure 2-4**

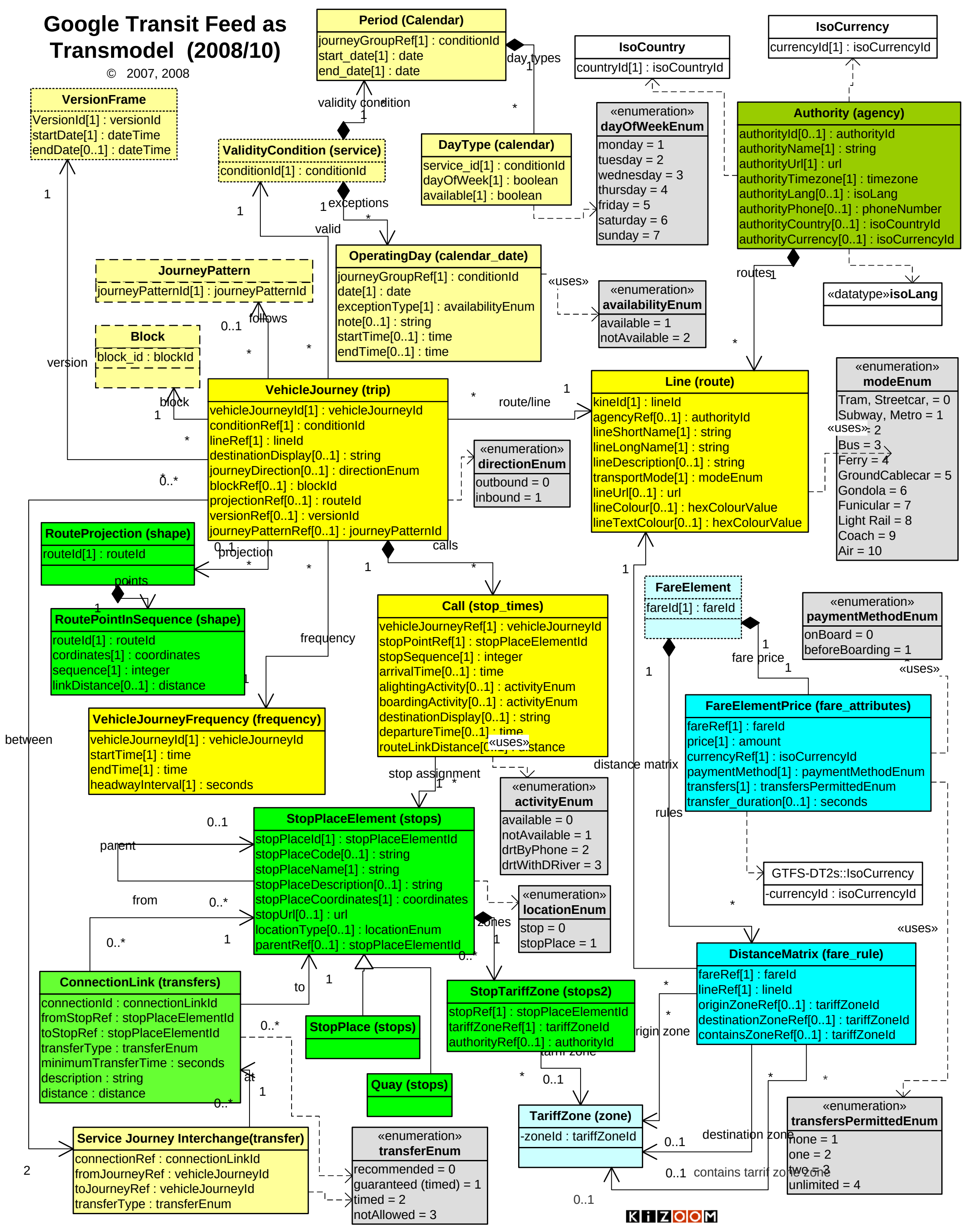
Google Transit Feed as Transmodel (2008/10)
© 2007, 2008
Period (Calendar)
journeyGroupRef[1] : conditionId
start_date[1] : date
end_date[1] : date
IsoCountry
countryId[1] : isoCountryId
IsoCurrency
currencyId[1] : isoCurrencyId
VersionFrame
VersionId[1] : versionId
startDate[1] : dateTime
endDate[0..1] : dateTime
ValidityCondition (service)
conditionId[1] : conditionId
DayType (calendar)
service_id[1] : conditionId
dayOfWeek[1] : boolean
available[1] : boolean
«enumeration» dayOfWeekEnum
monday = 1
tuesday = 2
wednesday = 3
thursday = 4
friday = 5
saturday = 6
sunday = 7
Authority (agency)
authorityId[0..1] : authorityId
authorityName[1] : string
authorityUrl[1] : url
authorityTimezone[1] : timezone
authorityLang[0..1] : isoLang
authorityPhone[0..1] : phoneNumber
authorityCountry[0..1] : isoCountryId
authorityCurrency[0..1] : isoCurrencyId
day types
validity condition
exceptions
valid
OperatingDay (calendar_date)
journeyGroupRef[1] : conditionId
date[1] : date
exceptionType[1] : availabilityEnum
note[0..1] : string
startTime[0..1] : time
endTime[0..1] : time
«uses»
«enumeration» availabilityEnum
available = 1
notAvailable = 2
routes
«datatype»isoLang
JourneyPattern
journeyPatternId[1] : journeyPatternId
Block
block_id : blockId
follows
version
block
VehicleJourney (trip)
vehicleJourneyId[1] : vehicleJourneyId
conditionId[1] : conditionId
lineRef[1] : lineId
destinationDisplay[0..1] : string
journeyDirection[0..1] : directionEnum
blockRef[0..1] : blockId
projectionRef[0..1] : routeId
versionRef[0..1] : versionId
journeyPatternRef[0..1] : journeyPatternId
route/line
«enumeration» directionEnum
outbound = 0
inbound = 1
Line (route)
lineId[1] : lineId
agencyRef[0..1] : authorityId
lineShortName[1] : string
lineLongName[1] : string
lineDescription[0..1] : string
transportMode[1] : modeEnum
lineUrl[0..1] : url
lineColour[0..1] : hexColourValue
lineTextColour[0..1] : hexColourValue
«enumeration» modeEnum
Tram, Streetcar, = 0
Subway, Metro = 1
Rail = 2
Bus = 3
Ferry = 4
GroundCablecar = 5
Gondola = 6
Funicular = 7
Light Rail = 8
Coach = 9
Air = 10
RouteProjection (shape)
routeId[1] : routeId
projection
points
calls
RoutePointInSequence (shape)
routeId[1] : routeId
cordinates[1] : coordinates
sequence[1] : integer
linkDistance[0..1] : distance
frequency
Call (stop_times)
vehicleJourneyRef[1] : vehicleJourneyId
stopPlaceRef[1] : stopPlaceElementId
stopSequence[1] : integer
arrivalTime[0..1] : time
alightingActivity[0..1] : activityEnum
boardingActivity[0..1] : activityEnum
destinationDisplay[0..1] : string
departureTime[0..1] : time
routeLinkDistance[0..1] : distance
FareElement
fareId[1] : fareId
«enumeration» paymentMethodEnum
onBoard = 0
beforeBoarding = 1
fare price
VehicleJourneyFrequency (frequency)
vehicleJourneyId[1] : vehicleJourneyId
startTime[1] : time
endTime[1] : time
headwayInterval[1] : seconds
FareElementPrice (fare_attributes)
fareRef[1] : fareId
price[1] : amount
currencyRef[1] : isoCurrencyId
paymentMethod[1] : paymentMethodEnum
transfers[1] : transfersPermittedEnum
transfer_duration[0..1] : seconds
between
stop assignment
distance matrix
«enumeration» activityEnum
available = 0
notAvailable = 1
drtByPhone = 2
drtWithDriver = 3
rules
StopPlaceElement (stops)
stopPlaceId[1] : stopPlaceElementId
stopPlaceCode[0..1] : string
stopPlaceName[1] : string
stopPlaceDescription[0..1] : string
stopPlaceCoordinates[1] : coordinates
stopUrl[0..1] : url
locationType[0..1] : locationEnum
parentRef[0..1] : stopPlaceElementId
parent
from
GTFS-DT2s::IsoCurrency
-currencyId : isoCurrencyId
«enumeration» locationEnum
stop = 0
stopPlace = 1
zones
DistanceMatrix (fare_rule)
fareRef[1] : fareId
lineRef[1] : lineId
originZoneRef[0..1] : tariffZoneId
destinationZoneRef[0..1] : tariffZoneId
containsZoneRef[0..1] : tariffZoneId
ConnectionLink (transfers)
connectionId : connectionLinkId
fromStopRef : stopPlaceElementId
toStopRef : stopPlaceElementId
transferType : transferEnum
minimumTransferTime : seconds
description : string
distance : distance
to
StopPlace (stops)
Quay (stops)
StopTariffZone (stops2)
stopRef[1] : stopPlaceElementId
tariffZoneRef[1] : tariffZoneId
authorityRef[0..1] : authorityId
tariff zone
origin zone
at
TariffZone (zone)
-zoneId : tariffZoneId
destination zone
contains tarrif zone
«enumeration» transfersPermittedEnum
none = 1
one = 2
two = 3
unlimited = 4
Service Journey Interchange(transfer)
connectionRef : connectionLinkId
fromJourneyRef : vehicleJourneyId
toJourneyRef : vehicleJourneyId
transferType : transferEnum
«enumeration» transferEnum
recommended = 0
guaranteed (timed) = 1
timed = 2
notAllowed = 3
KIZOOM

Figure 2-5 Detailed model of GTFS in Transmodel

2.3.2.1 Attribute Refinements

Minor points for possible discussion:

- The GTFS-*route_type* / Transmodel MODES should be refined. Coach is distinct from bus, and Light Rail/Suburban rail is distinct from tram/street car.
- An AUTHORITY can indicate a default currency type to use on all fares unless overridden.
- Some additional attributes are suggested for OPERATING DAY. (i) A note so that an explanation can be given to user and (ii) a start and end time so that a day boundary other than midnight can be specified. (E.g. 2am to 2am).
- A version frame is suggested for Vehicle Journey (And indeed all other top level elements).

2.3.2.2 Encoding points

- Use of the GML ***coordinates*** attribute could be used to encode lat & Lon. This more general and would allow for other coordinate systems.
- Intervals could be expressed using the XM Duration data type which supports hours minutes and second formats. One should clarify whether the Frequency ***headwayInterval*** is the minimum interval or the maximum interval.
- UTC time zones should be placed on individual times where relevant.
- The example uses lower camel case and avoids abbreviations in attribute names to strive for a literate coding style. This is just a matter of coding style: others are possible - e.g. retaining GTFS names exactly.
- A naming convention is used to distinguish the presence of identifiers as the unique identifiers of an entity, versus their presence as "foreign keys" to serialize a relationship. For example ***lineId*** (as the identifier of one Line) versus ***lineRef*** (for the relationships to ***Line*** from ***VehicleJourney*** and ***DistanceMatrix***).

2.3.3 Data types

The revised model uses similar limited number of data types and enumerations to GTFS. These are summarised in Figure 2-3.

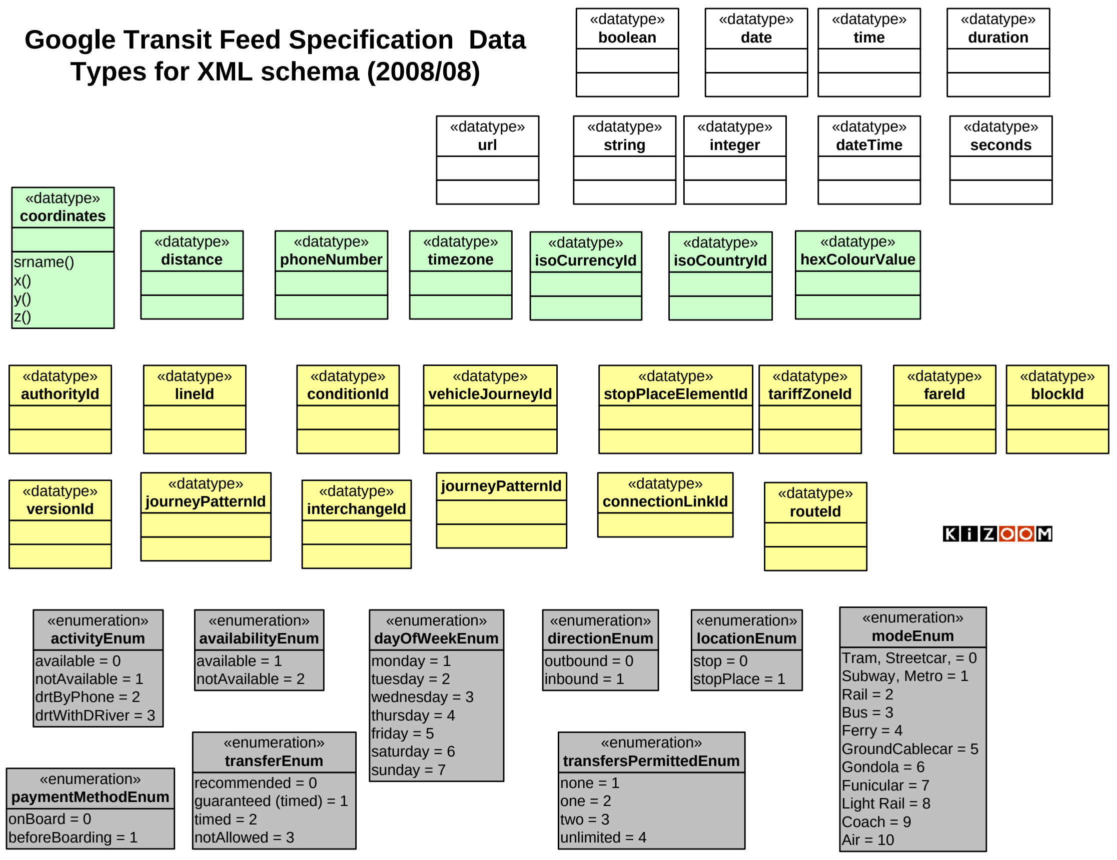

Figure 2-6 Data types for revised model

2.4 A Creating an XML schema for GTFS

In addition to expressing GTFS using Transmodel terminology, we propose creating an XML schema as an exact realisation of the model, using the same entity and attribute names.

Thus XML documents conforming to the schema can be used to contain any or all of the GTFS main entity types (GTFS-agency/AUTHORITY, GTFS-stop/STOP PLACE, GTFS-trip/VEHICLE JOURNEY) that can be independently defined. Documents can be validated against the schema using standard tools and a many different types of consistency checks applied.

2.4.1 Schema Hierarchy

In creating a Transmodel based XML schema for GTFS, containment can be used to group dependent child entities within a smaller number of top level elements (e.g. see Figure 2-7).

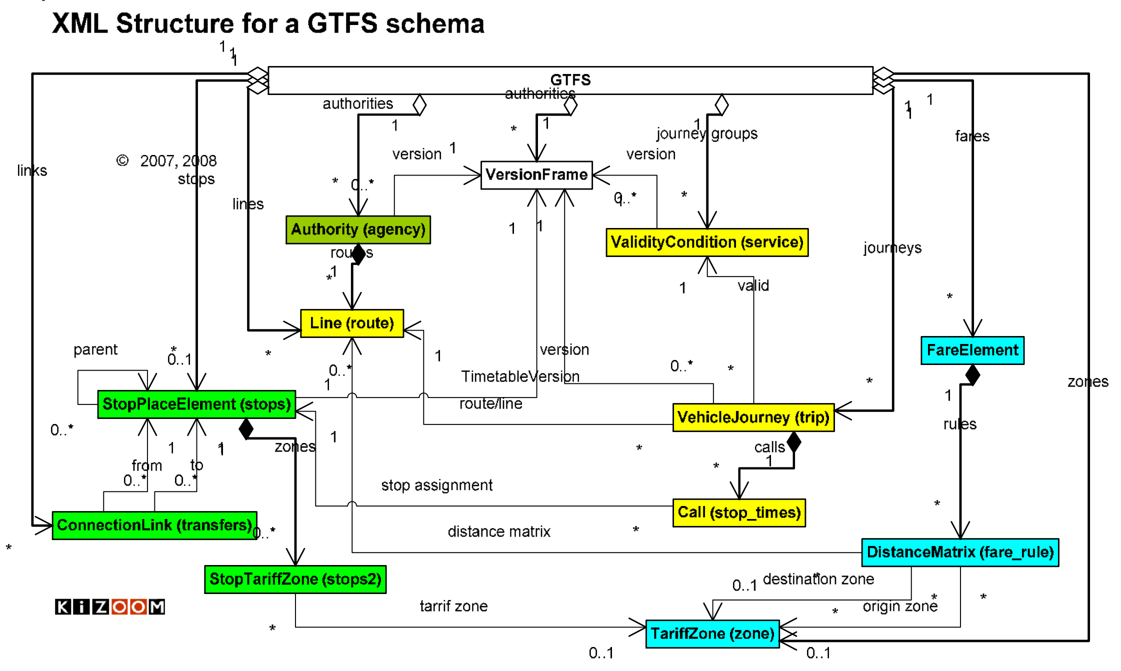

Figure 2-7 Outline of an GTFS XML schema

2.4.2 Example XML code fragment

The following example XML code fragment shows how the model could be rendered as an XML instance document

```
<Gtfs version=”1.0”>
            <StopPlaces>
                <StopPlace id=”2345001”>
                              <StopName>Gare de Nord</StopName>
                                        …..etc
                    <Quay id=”23450021”>
                              <StopName>Platform 5</StopName>
                                        …..etc
                    </Quay>
                </StopPlace>
                                        …..etc
            </StopPlaces>

            <VehicleJourneys>
                <VehicleJourneys id=”001” >
                <ValidityConditionRef=”C02”>
                    <Calls>
                    <Calls>
                              <StopPlaceRef>2345001</StopPlaceRef>
                              <StopSequence>01</StopSequence>
                              <DepartureTime>23:12:002Z</DepartureTime>
                              …..etc
                    </Call>
          </Calls>
                              …..etc
      </VehicleJourney>
                              …..etc

  </VehicleJourneys>
  <ValidityConditions>
      <VehicleJourneys id="001" >
      <ValidityConditionRef="C02">
  </ValidityConditions>

</Gtfs>
```

2.4.3 Schema Encoding

Creation of an XML schema is a relatively straightforward exercise which can be undertaken once the model is finalised.

- The schema should follow normal best practice for documentation, consistent naming conventions etc.
- The namespace should be well defined
- Reuse may be made of existing elements from Transmodel based schemas such as IFOPT and siri
- The schema should enforce self validating constraints such as data types, enumerations, keyrefs uniqueness, etc.
- The schema should be versioned.
- The data values should be assigned an explicit data instance version, e.g. using a ***VersionFrame*** entity and change dates.
- Containment can be indicated by using the relationship name as a wrapper tag, for example

```
          <VehicleJourney>
                    <Calls>
                              <Call>
                              …
                              <Call>
                              …..
                    </Calls
```

3 APPENDIX A -GTFS ANNOTATED WITH TRANSMODEL

This appendix lists the Google Transit Feed Specification, annotated where relevant with elements and attributes from Transmodel and IFOPT, and with a proposed name for each XML schema element.

3.1 GTFS-agency.txt (AUTHORITY)

3.1.1 Comments

A GTFS-*agency* corresponds to the Transmodel concept of AUTHORITY. The original version of GTFS supported one agency is allowed per file, i.e. there was no notion of multiple providers. It has now been refined to allow multiple agencies.

Transmodel in fact distinguishes between an OPERATOR and an AUTHORITY, and OPERATIONAL UNIT, allowing the attribution of services to operators in multi operator timetables (as say on UK Rail, or London Buses). However the AGENCY can be used for the authority for practical purposes.

Inclusion of a time-zone is needed in GTFS because GTFS-stop times don't use full UTC times (i.e. indicate time zone). Given the complex interlocking of European time zones we would propose using UTC for all purposes. Language and URL are useful presentation related attributes.

	Field Name	Use	Type	Description	TM Equivalent
V2	**agency_id**	**Optional, Unique**	agencyId (String)	Uniquely identifies a transit agency. A transit feed may represent data from more than one agency. The agency_id is dataset unique. This field is optional for transit feeds that only contain data for a single agency.	AUTHORITY Identifier ***authorityId***
V1	**agency_name**	**Required**	*string*	Name of the transit agency. Example TriMet	AUTHORITY NAME ***authorityName***
V1	**agency_url**	**Required**	*url*	Fully qualified URL for agency Example: http://www.trimet.org	***(agencyUrl)***
V1	**agency_-timezone**	**Required**	*time-zone*	Time zone where the transit agency is located. See http://en.wikipedia.org/wiki/List_of_tz_zones Example(s): America/Los_Angeles	***(authority-Timezone)***
V2	**agency_lang**	**Optional**	*lang*	two-letter ISO 639-1 code for the primary language used by this transit agency. This setting defines capitalization rules and other language-specific settings for all text contained in this transit agency's feed. Please refer to http://www.loc.gov/standards/iso639-2/php/code_list.php for a list of valid values.	***(authorityLang)***
V3	agency_-phone	**Optional**	*string*	A single voice telephone number for the specified agency. This field is a string value that presents the telephone number as typical for the agency's service area. It can and should contain punctuation marks to group the digits of the number. Dialable text (for example, TriMet's "503-238-RIDE") is permitted, but the field must not contain any other descriptive text.	***(authority-PhoneNumber)***

Table 3-1 GTFS Agency Table

3.2 GTFS-stops.txt Table (STOP PLACE)

3.2.1 Comments

The GTFS-*stops.txt* file provides a basic STOP PLACE model, similar to that of IFOPT.

Transmodel distinguishes between the SCHEDULED STOP POINT- the timetable reference to a stop - and the physical stop, which may be either a STOP PLACE (i.e. Station, pair of stops or other named grouping) or a specific point of access such as a pole, platform or gangway i.e. QUAY. (In IFOPT a QUAY may be further subdivided by a BOARDING POINT). A SCHEDULED STOP POINT can be assigned to a STOP PLACE and or QUAY - very often this assignment is assumed, that is implicit in the use of the same identifier for both the SCHEDULED STOP POINT and the STOP PLACE, but may be explicit and dynamic (e.g. in the case of a bus stop that moves temporarily, or train platform change.

Stops can have a complex relation to each other, e.g. in bus and train stations, and to cities and towns. Large stops such a Train platform may have a substructure (e.g. be used as separate platforms, or have boarding points). IFOPT has added to Transmodel a more elaborated representation of an interchange with distinct concepts of STOP PLACE, ACCESS SPACE, QUAY, and BOARDING POINT.

The original GTFS had no grouping mechanism for stops. Since 2008 stops may be linked to a single parent each, this allows a GTFS-stop to also be used as a STOP AREA, or STOP PLACE. (If it is used as a stop are it cannot be used as an access point as well).

The GTFS-stop has no mode (i.e. on cannot tell whether it is a bus or train).

There are thus two types of GTFS-*stop*, as station (i.e. AREA, or STOP PLACE.

GT FS	Field Name	Use	Type	Description	Equivalent
V1	**stop_id**	**Required, Unique**	stopId (String)	ID that uniquely identifies a stop. Multiple routes may use the same stop. **Example(s):** S81NATHIST	Identifier of STOP PLACE ELEMENT (will be Assigned to a SCHEDULED STOP POINT) ***stopPlaceId***
V3	stop_code	**Optional**	string	Contains short text or a number that uniquely identifies the stop for passengers. Stop codes are often used in phone-based transit information systems or printed on stop signage to make it easier for riders to get a stop schedule or real-time arrival information for a particular stop. The stop_code field should only be used for stop codes that are displayed to passengers. For internal codes, use stop_id. This field should be left blank for stops without a code.	***stopCode***
V1	**stop_name**	**Required**	string	**T**he name of a stop. A name that people will understand in the local and tourist vernacular. **Example(s):** 81 St-Museum of Natural History	***stopPlace-Name***

V1	**stop_desc**	**Required**	string	Description of a stop. Please provide useful, quality information. Do not simply duplicate the name of the stop. **Example(s):** The 81 St-Museum of Natural History stop is located at the southwest corner of the intersection at West 81st St. and Central Park West. The stop is two blocks south of the American Museum of Natural History.	***StoPlace-Description***
V1	**stop_lat**	**Required**	WGS 84 geodetic datum.	The latitude of a stop. The field value should contain a WGS 84 geodetic datum. **Example(s):** 40.781969	POINT ***coordinates***
V1	**stop_lon**	**Required**	WGS 84 geodetic datum.	The longitude of a stop. The field value should contain a WGS 84 geodetic datum. **Example(s):** 73.972011	POINT ***coordinates***
V1	**zone_id**	**Optional**	zoneId (String)	The fare zone for a stop. Zone IDs are required if you want to provide fare information using fare_rules.txt. **Example(s):**2	TARIFF ZONE ***tarrifZoneRef***
V2	**stop_url**	**Optional**	url	The URL of a web page about a particular stop. This should be different from the **agency_url** and the **route_url** fields.	*(stopUrl)*
V3	Location_type	**Optional**	enum	Identifies whether this stop ID represents a stop or station. If no location type is specified, or the location_type is blank, stop IDs are treated as stops. Stations may have different properties from stops when they are represented on a map or used in trip planning. The location type field can have the following values: ¤ **0** or blank - Stop. A location where passengers board or disembark from a transit vehicle. ¤ **1** - Station. A physical structure or area that contains one or more stop.	StopPlace
V3	Parent_station	**Optional**	string	. For stops that are physically located inside stations, the **parent_station** field identifies the station associated with the stop. To use this field, stops.txt must also contain a row where this stop ID is assigned location type=1.	parentStopPlaceElement

Table 3-2 GTFS Stops Table

This stop ID represents...	**This entry's location type...**	**This entry's parent_station field contains...**
A stop located inside a station.	0 or blank	The stop ID of the station where this stop is located.

The stop referenced by parent_station must have location_type=1.		
A stop located outside a station.	0 or blank	A blank value. The parent_station field doesn't apply to this stop.
A station.	1	A blank value. Stations can't contain other stations

3.3 GTFS-routes.txt (LINE / ROUTE)

3.3.1 *Comments*

The GTFS *routes.txt* table holds LINE attributes such as the name of the line or 'route'.

The GTFS-*route* entity is in effect a Transmodel LINE, (as opposed to a Transmodel ROUTE) - an arbitrary group of ROUTEs which is generally known to the public by a similar name or number. The GTFS-trips i.e. vehicle journeys of a LINE follow similar but not necessarily exactly the same service patterns.

Because it is concerned with computing and managing timetables and other data sets, Transmodel separates out the distinct information layers underpinning a vehicle journey. In particular, the infrastructure nodes and links (ROAD LEMENTS & RAIL ELEMENTS); the directional ROUTE, ROUTE LINKS & ROUTE POINTs over the infrastructure; the SERVICE PATTERN (the sequence of stops used when traversing a route in a particular direction, possibly as subset of the overall route); the JOURNEY PATTERN (the service pattern with timing information added) and VEHICLE JOURNEYs (An instance of a journey at a particular time which follows a JOURNEY PATTERN).

The Transmodel approach allows a large degree of reuse of elements and also preserves structural data that may be of use other applications, e.g. scheduling or AVL systems. Most of this is irrelevant to GTFS and the final schedule for passengers, but can be useful when fusing data sets.

At the service or route level the operational profile is described largely textually.

	Field Name	**Use**	***Type***	**Description**	**Equivalent**
V1	**route_id**	**Required, Unique**	routeId (String)	An ID that uniquely identifies a route. **Example(s):** R17X	LINE ***lineId***
V2	**agency_id**	**Optional**	*nmtoken*	an agency for the specified route. This value is referenced from the agency.txt file. Use this field when you are providing data for routes from more than one agency.	AUTHORITY ***authorityRef***
V1	**Route-_short_name**	**Required**	*string*	Short name of a route. This will often be the route number or route character(s). If the route does not have a short name, please use an empty string as the value for this field. **Example(s):** f the route full name is **17-NW 21st Ave/St Helens Rd**, then provide **17** as the **route_short_name** value.	LINE name ***lineShortName***
V1	**Route-_long_name**	**Required**	*string*	Full name of a route. This name will often include the route's destination or stop. If the route does not have a long name, please use an empty	LINE long name ***lineLongName***

				string as the value for this field. **Example(s):** If the route full name is **17-NW 21st Ave/St Helens Rd**, then please provide **NW 21st Ave/St Helens Rd** as the **route_long_name** value.	
V1	**route_desc**	**Optional**	*string*	A description of a route. Please provide useful, quality information. Do not simply duplicate the name of the route. **Example(s):** A trains operate between Inwood-207 St, Manhattan and Far Rockaway-Mott Avenue, Queens at all times. Also from about 6AM until about midnight, additional A trains operate between Inwood-207 St and Lefferts Boulevard (trains typically alternate between Lefferts Blvd and Far Rockaway.	LINE textual description ***description***
V1	**route_type**	**Required**	***0*** *- Tram* ***1*** *- Subway* ***2*** *- Rail* ***3*** *- Bus* ***4*** *- Ferry* *5 - CableCar* *6 - Gondola* *7- Funicular*	The type of transportation used on a route. Valid values for this field are: **Example(s):** 0 • **0** - Tram, Streetcar, Light rail. Any light rail or street level system within a metropolitan area. • **1** - Subway, Metro. Any underground rail system within a metropolitan area. • **2** - Rail. Used for intercity or long-distance travel. • **3** - Bus. Used for short- and long-distance bus routes. • **4** - Ferry. Used for short- and long-distance boat service. • **5** - Cable car. Used for street-level cable cars where the cable runs beneath the car. • **6** - Gondola, Suspended cable car. Typically used for aerial cable cars where the car is suspended from the cable. • **7** - Funicular. Any rail system designed for steep inclines. • **+++**	MODE ***mode*** • *European usage would Separate out light rail from Tram ?* • *Add Coach and AIR ??*
V2	**route_url**	**Optional**	***url***	contains the URL of a web page about that particular route. This	***(lineURL)***

				should be different from the **agency_url**. The value must be a fully qualified URL that includes **http://** or **https://**, and any special characters in the URL must be correctly escaped. See http://www.w3.org/Addressing/URL/4_URI_Recommentations.html for a description of how to create fully qualified URL values.	
V2	**route_color**	**Optional**	***hexValue***	defines a color that corresponds to a route. The color must be provided as a hexadecimal number. If the **route_color** makes overlaid text difficult to read, specify a contrasting text color with the **route_text_color** field.	***(lineColourURL)***
V2	**Route-_text_color**	**Optional**	***hexValue***	Can be used to specify a legible color to use for text drawn against a background of **route_color**.	***(lineText ColourURL***

Table 3-3 GTFS Route Table

3.4 Trips.txt (VEHICLE JOURNEY)

Th GTFS *trips* entity, describes and individual that is a scheduled journey of a vehicle. Note that in Transmodel the term TRIP is used for the itinerary of the passenger, not the vehicle.

The inclusion of ***Block*** allows journey planners to infer whether a change is needed on certain route topologies, e.g. circular routes.

The Shape element, corresponding to a TransXChange Route/Track allows the projection of a specific path on a map from a route.

	Field Name	**Use**	**Type**	**Description**	**TM**
V1	**trip_id**	**Required Unique**	tripId (String)	An ID that identifies a trip. **Example(s):**1AWE	VEHICLE JOURNEY identifier ***vehicleJourneyId /***
V1	**route_id**	**Required**	routeId (String)	ID that uniquely identifies a route. This value is referenced from the routes.txt file. **Example(s):** R17X	LINE identifier reference ***lineRef***
V1	**service_id**	**Required**	serviceId (String)	ID that uniquely identifies a set of dates when service is available for one or more routes. This value is referenced from the calendar.txt file. **Example(s):** WE	VALIDITY CONDITION identifier reference ***(conditionRef)***
V1	**Trip-_headsign**	**Optional**	string	The text that appears on a sign in the vehicle that identifies the trip's destination to passengers. **Example(s):** Montgomery Park	DESTINATION DISPLAY ***destinationDisplay***

V2	**Direction-_id**	**Optional**	directionEnum	**Optional**. The **direction_id** field contains a binary value that indicates the direction of travel for a trip. Use this field to distinguish between bi-directional trips with the same route_id. This field is not used in routing; it provides a way to separate trips by direction when publishing time tables. You can specify names for each direction with the **trip_headsign** field. • **0** - travel in one direction (e.g. outbound travel) • **1** - travel in the opposite direction (e.g. inbound travel) For example, you could use the trip_headsign and direction_id fields together to assign a name to travel in each direction on trip "1234", the trips.txt file would contain these rows for use in time tables: trip_id, ... , trip_headsign, direction_id 1234, ... , to Airport,0 1234, ... , to Downtown,1	DIRECTION ***direction***
V1	**block_id**	**Optional**	blockId (String)	The block to which the trip belongs. A block consists of two or more sequential trips made using the same vehicle, where a passenger can transfer from one trip to the next just by staying in the vehicle. The **block_id** is dataset unique. **Example(s):** B1AWE	BLOCK identifier reference ***blockRef***
V2	**shape_id**	**Optional**	shapeId (String)	an ID that defines a shape for the trip. This value is referenced from the shapes.txt file. The shapes.txt file allows you to define how a line should be drawn on the map to represent a trip.	ROUTE PROJECTION identifier reference ***routeRef***

Table 3-4 GTFS Trips Table

3.5 **GTFS-**stop_times.txt (STOP IN SEQUENCE | CALL)

3.5.1 Comments

The GTFS *stop_times* entity aggregates a STOP IN SEQUENCE for a VEHICLE JOURNEY, aggregating with the TIMETABLED PASSING TIMES and a stop activity. This is a common practical view; used for example by the SIRI ***Call*** element for both planned and estimated timetables.

The use of "local time" for the data type on times rather than UTC is unfortunate if there are cross timezone trips or variable summer time. We would propose using ITC on everything.

	Field Name	Use	Type	Description	TM
V1	**trip_id**	**Required, Unique**	tripId (String)	ID that identifies a trip. This value is referenced from the trips.txt file. **Example(s):** 1AWE	VEHICLE JOURNEY identifier ***vehicleJourney Ref***
V1	**arrival_time**	**Required**	HH:MM:SS local time	The arrival time at a specific stop for a specific trip on a route. The value should be expressed in HH:MM:SS local time after midnight of the day on which the trip begins. Please include all times for stops that are time points. Arrival times for the first and last stop in a trip are required. All other arrival times are optional and, if unavailable, may be represented with an empty string value. Stops without arrival times will be scheduled based on the nearest preceding timed stop. Do not interpolate stops **Example(s):** The following columns list stop times for a trip and the proper way to express those times in the **arrival_time** field: **Time** — **arrival_time value** 08:10:00 A.M. — 08:10:00 01:05:00 P.M. — 13:05:00 07:40:00 P.M. — 19:40:00 01:55:00 A.M. — 25:55:00 **Note:** Trips that span multiple dates will have stop times greater than **24:00:00**. For example, if a trip begins at 10:30:00 p.m. and ends at 2:15:00 a.m. on the following day, the stop times would be **22:30:00** and **26:15:00.** Entering those stop times as **22:30:00** and **02:15:00** would not produce the desired results.	PASSING TIME TIMETABLED ARRIVAL TIME ***arrivalTime***
V1	**departure_time**	**Required**	HH:MM:SS local time	The departure time from a specific stop for a specific trip on a route. The value should be expressed in HH:MM:SS local time after midnight of the day on which the trip begins. If the departure and arrival times are identical, please duplicate the values in the **arrival_time** and **departure_time** fields. **Example(s):**The following columns list stop times for a trip and the proper way to express	PASSING TIME TIMETABLED ARRIVAL TIME ***departureTime***

				those times in the **departure_time** field: **Time** / **departure_time value** 08:10:00 A.M. / 08:10:00 01:05:00 P.M. / 13:05:00 07:40:00 P.M. / 19:40:00 01:55:00 A.M. / 25:55:00	
V1	**stop_id**	**Required**	stopId (String)	An ID that uniquely identifies a stop. Multiple routes may use the same stop. This value is referenced from the stops.txt file. **Example(s):**S81NATHIST	STOP PLACE ***StopPointCode***
V1	**stop_sequence**	**Required**	integer	The cardinal number that identifies the order of a stop on a particular trip. The first stop on the trip should have a **stop_sequence** of **1**; the second stop on the trip should have a **stop_sequence** of **2**, and so forth. **Example(s):**3	SEQUENCE ***stopSequence***
V2	**Stop_headsign**	**Optional**	**string**	the text that appears on a sign that identifies the trip's destination to passengers. Use this field when the headsign changes between stops. If this headsign is associated with an entire trip, use trip_headsign instead. See a Google Transit Trip Planner screenshot highlighting the **headsign**.	DESTINATION DISPLAY
V1	**pickup_type**	**Optional**	**0** - Regularly scheduled pickup **1** - No pickup available **2** - Must phone agency to arrange pickup **3** - Must coordinate with driver to arrange pickup	Whether passengers are picked up at a stop as part of the normal schedule or whether a pickup at the stop is not available. This field also allows the transit agency to indicate that passengers must call the agency or notify the driver to arrange a pickup at a particular stop. Valid values for this field are: The default value for this field is **0**.	*Boarding Activity* ***boardingActivity***
V1	**drop_off_type**	**Required**	**0** - Regularly scheduled drop off **1** - No drop off	Whether passengers are dropped off at a stop as part of the normal schedule or whether a drop off at the stop is not available. This field also allows the transit agency to indicate that passengers must call the agency or notify the driver to	*Alighting Activity* ***alightingActivity?***

			available **2** - Must phone agency to arrange drop off **3** - Must coordinate with driver to arrange drop of	arrange a drop off at a particular stop. Valid values for this field are: The default value for this field is **0**.	
V2	**shape_dist_traveled**	**Optional**	**decimal**	Positions a stop as a distance from the first shape point. The **shape_dist_traveled** field represents a real distance traveled along the route in units such as feet or kilometers. For example, if a bus travels a distance of 5.25 kilometers from the start of the shape to the stop, the **shape_dist_traveled** for the stop ID would be entered as "5.25". The values used for **shape_dist_traveled** must increase along with **stop_sequence**: they cannot be used to show reverse travel along a route. The units used for **shape_dist_traveled** in the stop_times.txt file must match the units that are used for this field in the shapes.txt file.	LINK DISTANCE ***routeLinkDistance***

Table 3-5 GTFS Stop Times Table

3.6 GTFS-calendar.txt (VALIDITY CONDITION)

3.6.1 Comments

The GTFS-*calendar* specifies a PERIOD and DAY TYPE part of a Transmodel VALIDITY CONDITION - it allows GTFS-*Service* period and day types to be specified. Thus a separate SERVICE is needed to associate a given calendar with a GTFS-Trip or trips.

- Service dates are assumed to start and end at midnight.

Field Name	**Use**	**Type**	**Description**	**Equivalent**
service_id	**Required, Unique**	nmtoken	An ID that uniquely identifies a set of dates when service is available for one or more routes. **Example(s):** WE	VALIDITY CONDITION ***conditionId***
monday	**Required**	Binary o/1	Whether the service is valid for all Mondays. **Note:** may list exceptions for particular dates, such as holidays, in the calendardates.txt file.	DAY TYPE / DAY OF WEEK
tuesday	**Required**	Binary o/1	Whether the service is valid for all Tuesdays.	DAY TYPE / DAY OF WEEK

wednesday	**Required**	Binary o/1	Whether the service is valid for all Wednesdays.	DAY TYPE / DAY OF WEEK
thursday	**Required**	Binary o/1	Whether the service is valid for all Thursdays.	DAY TYPE / DAY OF WEEK
friday	**Required**	Binary o/1	Whether the service is valid for all Fridays.	DAY TYPE / DAY OF WEEK
saturday	**Required**	Binary o/1	Whether the service is valid for all Saturdays.	DAY TYPE / DAY OF WEEK
sunday	**Required**	Binary o/1	Whether the service is valid for all Sundays.	DAY TYPE / DAY OF WEEK
start_date	**Required**	YYYYMMDD	Start date for the service.	PERIOD ***startDate***
end_date	**Required**	YYYYMMDD	End date for the service. This date is included in the service interval.	PERIOD ***endDate***

Table 3-6 GTFS Calendar Table

3.7 GTFS-calendar_dates.txt (VALIDITY CONDITION)

3.7.1 Comments

The GTFS- *calendar_dates* table allows OPERATING DAYs to be specified as part of the VALIDITY CONDITION (ie GTFS-service). In particular it allows public holidays and exceptions to the regular day types to be expressed as specific calendar dates on which the service does or does not run. There are no public holiday day types, thus exceptions can be stated, but they must be explicitly repeated as dates each year. There is no text associated with the date so exceptions cannot be explained to the user e.g. as "Christmas day" or "Holiday schedule" - this has to be done in the textual service description.

Field Name			**Details**	
service_id	**Required**	serviceId (String)	ID that uniquely identifies a set of dates when service is available for one or more routes. **Example(s):** WE	VALIDITY CONDITION identifier ***conditionId*** l
date	**Required**	YYYYMMDD	a particular date when service availability is different than the norm as indicated by the **exception_type**	DAY TYPE date ***date***
exception_type	**Required**	**exceptionEnum** **1** - service has been added for the specified date. **2** - service has been removed for the specified date.	Indicates whether service is available on the date specified in the **date** field. For example, suppose a route has one set of trips available on holidays and another set of trips available on all other days. You could have one **service_id** that corresponds to the regular service schedule and another **service_id** that corresponds to the holiday schedule. For a particular holiday, you would use the calendar_dates file to add the holiday to the	DAY TYPE attribute ***exceptionType***

			holiday **service_id** and to remove the holiday from the regular **service_id** schedule.	

Table 3-7 GTFS Calendar Dates

3.8 GTFS-fare_attributes.txt (FARE PRODUCT)

3.8.1 *Comments*

GTFS *Fare attributes* allows the expression of a basic fare product with simple FARE USAGE PARAMETERS and PAYMENT METHODS.

This is a very basic PRICE GROUP compared with the many Fare models considered in Transmodel and the recent UK FareXchange study.

Field Name	Use	Type	Description	Equivalent
fare_id	**Required, Unique**	fareId (String)	ID that uniquely identifies a fare class. **Example(s):** b	FARE ELEMENT identifier
price	**Required**	Currency amount	**Required**. The **price** field contains the fare price, in the unit specified by **currency_type.** **Example(s):** 1.75	FARE PRICE
currency_type	**Required**	ISO 4217 Currency type	The currency used to pay the fare. ISO 4217 alphabetical currency code http://www.iso.org/iso/en/prods-services/popstds/currencycodeslist.html. **Example(s):** USD	(currency)
payment_method	**Required**	**methodEnum** **0** - Fare is paid on board. **1** - Fare must be paid before boarding.	When the fare must be paid. Valid values for this field are:	PAYMENT METHOD ***oayment-Method***
transfers	**Required**	**0** - No transfers permitted on this fare. **1** - Passenger may transfer once. **2** - Passenger may transfer twice. **(empty)** - If this field is empty,	The number of transfers permitted on this fare.	FARE PARAMETER attrinute (***rransfers***)

		unlimited transfers are permitted.		
transfer_duration	**Optional**	seconds	The length of time in seconds before a transfer expires. **Example(s):** 4000	FARE PARAMETER attribute (***transfer-Duration***)

Table 3-8 GTFS Fare attributes Table

3.9 GTFS-fare_rules.txt (DISTANCE MATRIX)

3.9.1 Comments

The GTFS-*fare_ rules* allow a simple zonal TARIFF STRUCTURE to be expressed. Different fares for tariff zone to tariff zone fares can be stated - this corresponds to a Transmodel DISTANCE MATRIX and PRICE TABLE of Transmodel.

This is a relatively trivial subset from the many Fare models supported in Transmodel and considered in the recent UK FareXchange study.

GTFS thus has a basic tariff zone structure, but not stage fares - and there is no way to specify vehicle journey level availability conditions - or complex Fare Product usage rules as are found for UK rail.

	Field Name	**Use**	**Type**	**Description**	**Equivalent**
V1	**fare_id**	**Required, Unique**	fareId (String)	ID that uniquely identifies a fare class. This value is referenced from the fare_attributes.txt file. **Example(s):**b	FARE STRUCTURE ELEMENT identifier ***fareId***
V1	**route_id**	**Optional**	routeId (String)	Route associated with the fare. Route IDs are referenced from the routes.txt file. If you have several routes with the same fare attributes, create a row in fare_rules.txt for each route. For example, if fare class "b" is valid on route "TSW" and "TSE", the fare_rules.txt file would contain these rows for the fare class: b,TSW b,TSE **Example(s):**TSW	LINE identifier reference ***lineRef***
V1	**origin_id**	**Optional**	zoneId (String)	Origin zone ID associated with the fare. Zone IDs are referenced from the stops.txt file. If you have several origin IDs with the same fare attributes, create a row in fare_rules.txt for each origin ID. For example, if fare class "b" is valid for all travel originating from either zone "2" or zone "8", the fare_rules.txt file would contain these rows for the fare class: b, , 2 b, , 8 **Example(s):** 2	TARIFF ZONE - DISTANCE MATRIX identifier reference ***originRef***

V1	**destination_id**	**Optional**	zoneId (String)	Destination zone ID associated with the fare. Zone IDs are referenced from the stops.txt file. If you have several destination IDs with the same fare attributes, create a row in fare_rules.txt for each destination ID. For example, you could use the origin_Id and destination_Id fields together to specify that fare class "b" is valid for travel between zones 3 and 4, and for travel between zones 3 and 5, the fare_rules.txt file would contain these rows for the fare class: b, , 3,4 b, , 3,5 **Example(s):** 4	TARIFF ZONE - DISTANCE MATRIX identifier reference ***destinationRef***
V1	**contains_id**	**Optional**	zoneId (String)	Associates the fare ID with all routes that pass through a specified location. The contains ID field is a zone ID, referenced from the stops.txt file. For example, if fare class "c" is associated with all travel on the GRT route that passes through zone 6, the fare_rules.txt would contain this row: c,GRT,,,6 **Example(s):** 6	TARIFF ZONE - identifier reference (***containsRef***)

Table 3-9 GTFS Fare Rules Table

3.10 GTFS-shapes.txt ( )

3.10.1 Comments

The GTFS-shape allows the LINK PROJECTION of an individual VEHICLE JOURNEY on to a map LINK PROJECTION.

	Field Name	Use	Type	Description	Equivalent
V2	**shape_id**	**Required**	shapeId (String)	An ID that uniquely identifies a shape.	ROUTE PROJECTION identifier ***Route_ProjectionId***
V2	**shape_pt_lat**	**Required**	lat	Associates a shape point's latitude with a shape ID. The field value must be a valid WGS 84 latitude. Each row in shapes.txt represents a shape point in your shape definition. For example, if the shape "A_shp" has three points in its definition, the shapes.txt file would contain these rows to define the shape: A_shp,37.61956,122.48161,1 A_shp,37.64430,122.41070,2 A_shp,37.65863,122.30839,3	POINT location attribnute ***coordinates***
V2	**shape_pt_lon**	**Required**	log	Associates a shape point's longitude with a shape ID. The field value must be a valid WGS 84 longitude. Each row	POINT location attribnute

				in shapes.txt represents a shape point in your shape definition. For example, if the shape "A_shp" has three points in its definition, the shapes.txt file would contain these rows to define the shape: A_shp,37.61956,122.48161,1 A_shp,37.64430,122.41070,2 A_shp,37.65863,122.30839,3	***coordinates***
V2	**shape_pt-_sequence**	**Required**	integer	Associates the latitude and longitude of a shape point with its sequence order along the shape. The first shape point in the shape definition should have a **shape_pt_sequence** of **1**, the second shape point should have a **shape_pt_sequence** of **2**, and so forth. You must use integer values.	LINK SEQUENCE ***sequence***
V2	**shape_dist-_traveled**	**Optional**	distance	When used in the shapes.txt file, the **shape_dist_traveled** field positions a shape point as a distance traveled along a shape from the first shape point. The **shape_dist_traveled** field represents a real distance traveled along the route in units such as feet or kilometers. The values used for **shape_dist_traveled** must increase along with **shape_pt_sequence**: they cannot be used to show reverse travel along a route. The units used for **shape_dist_traveled** in the shapes.txt file must match the units that are used for this field in the stop_times.txt file. For example, if a bus travels along the three points defined above for A_shp, the additional **shape_dist_traveled** values (shown here in kilometers) would look like this: A_shp,37.61956,122.48161,1,0 A_shp,37.64430,122.41070,2,6.8310 A_shp,37.65863,122.30839,3,15.8765	ROUTE LINK / DISTANCE ***linkDistance***

3.11 GTFS-frequency.txt ( )

3.11.1 Comments

GTFT-frequency is used to describe frequency based services. It allows a frequency to be specified, effectively generating a timetable from a single Vehicle journey.

This is similar to the UK TransXChange ***Frequency*** Element, used to describe a service that occurs at a specified interval within a time window rather than running to an exact timetable.

	Field Name	Use	Type	Description	Equivalent
V2	**trip_id**	**Required**	tripId (String)	An ID that identifies a trip on which the specified frequency of service applies. Trip IDs are referenced from the trips.txt file.	VEHICLE JOURNEY identifier reference ***vehicle-JourneyRef /***
V2	**start_time**	**Required**	time	The time at which service begins with the specified frequency. For times occurring after midnight, enter the time as a value greater than 24:00:00 in HH:MM:SS local time for the day on which the trip schedule begins. E.g. 25:35:00.	PERIOD end ***startTime***
V2	**end_time**	**Required**	time	The time at which service changes	PERIOD start

				to a different frequency (or ceases). For times occurring after midnight, enter the time as a value greater than 24:00:00 in HH:MM:SS local time for the day on which the trip schedule begins. E.g. 25:35:00.	***endTime/***
V2	**headway_secs**	**Required**	secs	The time between departures from the same stop (headway) for this trip type, during the time interval specified by **start_time** and **end_time**. The headway value must be entered in seconds.	***frequency Interval***

3.12 GTFS-transfers.txt ( )

3.12.1 Comments

The transfers file allows the specification of interchange rules. See discussion on interchange for limitations of this model.

	Field Name	Use	Type	Description	Equivalent
V3	**from_stop_id**	**Required**	stopId (String)	stop ID that identifies a stop or station where a connection between routes begins. Stop IDs are referenced from the stops.txt file. If the stop ID refers to a station that contains multiple stops, this transfer rule applies to all stops in that station.	CONNECTION LINK from stop ***fromStopRef***
V3	**to_stop_id**	**Required**	stopId (String)	a stop ID that identifies a stop or station where a connection between routes ends. Stop IDs are referenced from the stops.txt file. If the stop ID refers to a station that contains multiple stops, this transfer rule applies to all stops in that station.	CONNECTION LINK to stop ***toStopRef***
V3	**transfer_type**	**Required**	time	specifies the type of connection for the specified (from_stop_id, to_stop_id) pair. Valid values for this field are: • **0** or **(empty)** - This is a recommended transfer point between two routes. • **1** - This is a timed transfer point between two routes. The departing vehicle is expected to wait for the arriving one, with sufficient time for a passenger to transfer between routes • **2** - This transfer requires a minimum amount of time between arrival and departure to ensure a connection. The time required to transfer is specified by **min_transfer_time.** **3** - Transfers are not possible between routes at this location.	ADVERTISED, PLANNED, GUARANTEED INTERCHANGE ***transferType)***

V3	**min_transfer_time**	**Optional**	secs	When a connection between routes requires an amount of time between arrival and departure (transfer_type=2), the **min_transfer_time** field defines the amount of time that must be available in an itinerary to permit a transfer between routes at these stops. The min_transfer_time must be sufficient to permit a typical rider to move between the two stops, including buffer time to allow for schedule variance on each route. The min_transfer_time value must be entered in seconds, and must be a non-negative integer.	SERVICE JOURNEY INTERCHANGE Duration ***minimumTransferTime***